

Руководство администратора

Универсальная программная платформа для корпоративных Информационных систем ПАО «РусГидро»

2025 г.

Оглавление

1 Общие сведения 5
2 Условия работы Системы 6
2.1 Программные требования к Системе 6
2.2 Аппаратные требования к Системе 8
3 Сборка и развертывание интеграционных решений Enterprise Integrator 11
3.1 Сборка приложения Enterprise Integrator из исходных текстов 11
3.2 Конфигурирование DSS сервисов ESB профиля 13
3.3 Установка артефактов на серверы Enterprise Integrator 15
4 Управление Carbon приложениями Enterprise Integrator 17
4.1 Описание Carbon 17
4.1.1 Системные службы и пользовательский интерфейс 17
4.1.2 Смена базы данных 18
4.1.3 Порты по умолчанию 19
5 Консоль администрирования Enterprise Integrator 21
5.1 Вкладка: «Домашняя» - «Управлять» 22
5.1.1 Раздел «Сервисы» 22
5.1.2 Раздел «Сервисная шина» 28
5.1.3 Раздел «Коннекторы» 42
5.1.4 Раздел «Средства безопасного Хранения» 43
5.1.5 Раздел «Приложения Carbon» 43
5.1.6 Раздел «Модули» 46
5.1.7 Раздел «Разделы» 47
5.1.8 Раздел «Выключение/Перезапуск» 48
5.1.9 Раздел «Реестр» 49
5.1.10 Вкладка: «Мониторинг» 52
5.1.11 «Журналы приложений» 53
5.1.12 «Системная статистика » 54
5.2 Мониторинг SOAP сообщений. Включение статистики / отслеживания сообщений. 56
5.2.1 «Системные журналы» 58
5.2.2 «Потоки сообщений» 59
5.3 Вкладка: «Настройки» 60
5.3.1 Раздел «Пользователи и Роли» 60
5.3.2 Раздел «Пользовательские хранилища» 62
5.3.3 Раздел «Хранилище ключей» 64
5.3.4 «Логирование» 66
5.3.5 «Источники данных» 67
5.3.6 «Приемник событий» 69
5.3.7 «Роли сервера» 70
6 Администрирование API Manager 72
6.1 Управление пользователями и ролями 72
6.1.1 Настройка хранилища пользователей. Подключение к Active Directory 72
6.1.2 Добавление роли 74
6.2 Создание областей действия API 74
6.3 Создание API 76
6.3.1 Подготовка WSDL-описания для API 76
6.3.2 Создание API 76
7 Администрирование Apache Kafka 77
7.1 Администрирование Apache Kafka 77

7.1.1 Настройка и установка сертификатов 77
7.1.2 Аутентификация Zookeeper 78
7.1.3 Аутентификация Kafka 78
7.2 Управление топиками. 80
7.2.1 Создание топика 80
7.2.2 Настройка доступа (ACL) 80
7.3 Основные операции с топиками 80
8 Администрирование Apache Superset 82
8.1 Доступ к компонентам Системы 82
8.2 Мониторинг работы компонентов Системы 83
8.3 Логирование компонентов Системы 84
8.4 Настройка ролей 84
8.5 Пользовательские настройки Системы 86
8.6 Переименование колонок 87
8.7 Остановка и запуск 88
9 Администрирование Apache NiFi 89
9.1 Доступ к компонентам Системы 89
9.2 Мониторинг работы компонентов Системы 89
9.3 Логирование компонентов Системы 90
10 Администрирование OpenMetadata 93
10.1 Доступ к компонентам Системы 93
10.2 Назначение ролей 94
10.3 Мониторинг работы компонентов Системы 97
10.4 Логирование компонентов Системы 97
11 Администрирование ClickHouse 99
11.1 Доступ к компонентам Системы 99
11.2 Мониторинг работы компонентов Системы 99
11.3 Логирование компонентов Системы 100
12 Администрирование KeyCloak 101
12.1 Доступ к компонентам Системы 101
12.2 Управление областями 102
12.3 Мониторинг работы компонентов Системы 106
12.4 Логирование компонентов Системы 106
13 Администрирование Apache Airflow 107
13.1 Доступ к компонентам Системы 107
13.2 Мониторинг работы компонентов Системы 108
13.3 Логирование компонентов Системы 108
14 Администрирование Grafana Loki 109
14.1 Доступ к компонентам подсистемы 109
14.2 Управление пользователями Grafana 110
14.3 Мониторинг и управление компонентами подсистемы 111
14.4 Просмотр собираемых из систем логов через Grafana 111
14.5 Логирование компонентов подсистемы через Docker 112
15 Администрирование TestY 113
15.1 Доступ к компонентам подсистемы 113
15.2 Управление пользователями TestY 114
15.3 Мониторинг и управление компонентами подсистемы 115
15.4 Логирование компонентов подсистемы 115
16 Администрирование JSReport 116
16.1 Доступ к компонентам подсистемы 116
16.2 Управление пользователями JSReport 117

16.3 Мониторинг и управление компонентами подсистемы 118
16.4 Логирование компонентов подсистемы 118
17 Резервное копирование баз данных и компонентов Системы 120
17.1 Общие рекомендации к резервированию 120
17.2 Резервное копирование и восстановление данных ClickHouse 121
17.2.1 Установка утилиты clickhouse-backup 121
17.2.2 Настройка clickhouse-backup через config файл 121
17.2.3 Создание резервной копии 122
17.2.4 Восстановление данных из резервной копии 123
17.2.5 Создание резервной копии по расписанию через crontab 124
17.3 Резервное копирование и восстановление данных OpenMetadata 124
17.3.1 Требования 124
17.3.2 Установка 124
17.3.3 Предварительные работы 125
17.3.4 Создание резервной копии через backup CLI 125
17.3.5 Восстановление Metadata БД 126
18 Настройки производительности системы 128
18.1 Настройка производительности на уровне ОС 128
18.2 Настройки JVM 130
18.3 Конфигурация кэширования 130

1 Общие сведения

Руководство администратора содержит описание технической архитектуры, структуры базы данных и компонентов, и предназначено для помощи в эксплуатации Единой интеграционной платформы ООО «РусГидро ИТ сервис» (далее - Система).

Система предназначена для:

- обеспечения интеграции и обмена данными между различными корпоративными приложениями;
- извлечения, преобразования и загрузки данных из разных источников в хранилище для анализа;
- формирования отчетности для предоставления бизнес-пользователям актуальных и структурированных данных для принятия решений.

Исходя из перечня необходимого программного обеспечения, для полноценной работы Системы необходимы следующие типы администраторов:

1. Системный администратор;
2. Администратор базы данных.

2 Условия работы Системы

2.1 Программные требования к Системе

Система представляет собой комплексное программное и аппаратное обеспечение, объединяющее несколько взаимосвязанных подсистем, каждая из которых выполняет определенные функции.
Система состоит из следующих модулей:

1. WSO2 Enterprise Integrator: Модуль для интеграции различных приложений и сервисов, обеспечивающий маршрутизацию, трансформацию и надежную передачу сообщений между системами;
2. WS2 API Manager: Модуль для управления API, включающий функции публикации, мониторинга, обеспечения безопасности и аналитики для API;
3. Apache Kafka: Модуль для потоковой передачи данных в реальном времени, обеспечивающий надежное и масштабируемое хранение и обработку сообщений;
4. Apache Superset: Модуль для анализа данных и создания интерактивных визуализаций, предоставляющий пользователю удобный интерфейс для создания и просмотра дашбордов, отчетов и графиков;
5. Apache NiFi: Модуль управления потоками данных, который обеспечивает автоматизацию и управление передачей, преобразованием и агрегацией данных из различных источников в системе;
6. ClickHouse: Модуль хранения и обработки аналитических данных, предназначенный для быстрой и эффективной обработки больших объемов данных;
7. Веб-приложение FlowGraph: Модуль пользовательского интерфейса, представляющий собой веб-приложение, которое позволяет пользователям взаимодействовать с графом, выполнять запросы и просматривать результаты через удобный пользовательский интерфейс;
8. Keycloak: Модуль для обеспечения безопасности и управления доступом, который предоставляет функциональность аутентификации и авторизации пользователей, а также управление ролями и правами доступа к системе и её компонентам;
9. OpenMetadata: Модуль для управления метаданными, который обеспечивает возможности по управлению, документированию и исследованию метаданных в Системе;
10. ZooKeeper: Модуль для обеспечения координации и управления распределенными сервисами и компонентами Системы, обеспечивая синхронизацию и обнаружение сервисов;
11. Tantor: Модуль для обеспечения реляционной базы данных, который используется для хранения метаданных, управления пользователями и других операций, требующих постоянного хранения данных;
12. Apache Airflow: Модуль для для разработки, планирования и мониторинга рабочих процессов ETL;
13. Grafana Loki: Модуль для централизованного сбора, хранения и анализа логов, обеспечивающий удобный поиск, фильтрацию и визуализацию сообщений

журналов из различных систем через интерфейс Grafana; поддерживает масштабируемую архитектуру и интеграцию с другими инструментами мониторинга;

14. TestY: Модуль для управления процессом ручного тестирования, обеспечивающий создание и организацию тест-кейсов; фиксацию результатов и отслеживание статуса тестов; поддерживает управление проектами и тестовыми наборами, а также формирование отчетности по результатам тестирования;
15. JSReport: Модуль для генерации отчетов на основе JavaScript, позволяющий формировать динамические документы (PDF, HTML, Excel, Word и др.) через шаблоны, API и встроенные движки шаблонизации; обеспечивает управление шаблонами, интеграцию с внешними данными и расширяемость через плагины.

2.2 Аппаратные требования к Системе

Таблица 1

Назначение	CPU cores	RAM, Гб	SSD, Гб	HDD2, Гб	ОС	СУБД и приложения
Prod - WSO2 Enterprise Integrator 01	8	24	-	45	AlterOS 7.5	ssh, EI
Prod - WSO2 Enterprise Integrator 02	8	24	-	45	AlterOS 7.5	ssh, EI
Prod - Брокер сообщений 01	8	8	500	-	AlterOS 7.5	ssh, Kafka Message Broker
Prod - Брокер сообщений 02	8	8	500	-	AlterOS 7.5	ssh, Kafka Message Broker
Prod - Брокер сообщений 03	8	8	500	-	AlterOS 7.5	ssh, Kafka Message Broker
Prod - СУБД	4	16	-	500	AlterOS 7.5	ssh, Tantor 14
Prod - API менеджер 01	2	8	-	64	AlterOS 7.5	ssh, APIM
Prod - API менеджер 02	2	8	-	64	AlterOS 7.5	ssh, APIM
Prod - ClickHouse 01	8	128	100	500	AlterOS 7.5	ssh, yum-utils
Prod - ClickHouse 02	8	128	100	500	AlterOS 7.5	ssh, yum-utils
Prod - БД для сервисов FlowGraph, OpenMetadata, Superset	4	16	200		AlterOS 7.5	ssh, Tantor 14
Prod - Apache Nifi (ETL инструмент)	8	16	200		AlterOS 7.5	ssh, Java 8 или Java 11
Prod - Flow Graph (Веб приложение "Граф интеграций")	2	4		100	AlterOS 7.5	ssh, Docker 20.10, Nginx
Prod - KeyCloak (Авторизация\аутентификация для Flow Graph)	2	4		100	AlterOS 7.5	ssh, Java 8 или Java 11
Prod - Apache Superset (BI система)	8	16	200		AlterOS 7.5	ssh, Docker 20.10, Nginx
Prod - OpenMetadata (Каталог данных)	8	16	200		AlterOS 7.5	ssh, Docker 20.10, Nginx
Prod - Apache Airflow (ETL инструмент)	10	40	50		AlterOS 7.5	ssh, Docker 20.10
Prod - Grafana Loki	4	8	400		AlterOS 7.5	ssh, Docker 20.10

Prod - TestY (TMS инструмент)	4	8	50		AlterOS 7.5	ssh, Docker 20.10
Prod - JSReports	4	8	50		AlterOS 7.5	ssh, Docker 20.10
QA - WSO2 Enterprise Integrator 01	4	8	-	45	AlterOS 7.5	ssh, EI
QA - WSO2 Enterprise Integrator 02	4	8	-	45	AlterOS 7.5	ssh, EI
QA - Брокер сообщений 01	8	8	-	500	AlterOS 7.5	ssh, Kafka Message Broker
QA - Брокер сообщений 02	8	8	-	500	AlterOS 7.5	ssh, Kafka Message Broker
QA - Брокер сообщений 03	8	8	-	500	AlterOS 7.5	ssh, Kafka Message Broker
QA - СУБД	4	8	-	50	AlterOS 7.5	ssh, Tantor 14
QA - WSO2 API Manager 01	4	8	-	45	AlterOS 7.5	ssh, APIM
QA - WSO2 API Manager 02	4	8	-	45	AlterOS 7.5	ssh, APIM
QA - ClickHouse 01	4	64	100	500	AlterOS 7.5	ssh, yum-utils
QA - ClickHouse 02	4	64	100	500	AlterOS 7.5	ssh, yum-utils
QA - БД для сервисов FlowGraph, OpenMetadata, Superset	4	16	200		AlterOS 7.5	ssh, Tantor 14
QA - Apache Nifi (ETL инструмент)	8	16	200		AlterOS 7.5	ssh, Java 8 или Java 11
QA - Flow Graph (Веб приложение "Граф интеграций")	2	4		100	AlterOS 7.5	ssh, Docker 20.10, Nginx
QA - KeyCloak (Авторизация\аутентификация для Flow Graph)	2	4		100	AlterOS 7.5	ssh, Java 8 или Java 11
QA - Apache Superset (BI система)	8	16		150	AlterOS 7.5	ssh, Docker 20.10, Nginx
QA - OpenMetadata (Каталог данных)	8	16		150	AlterOS 7.5	ssh, Docker 20.10, Nginx
QA - Apache Airflow (ETL инструмент)	4	8	50		AlterOS 7.5	ssh, Docker 20.10
QA - Grafana Loki	2	4	50		AlterOS 7.5	ssh, Docker 20.10

QA - TestY (TMS инструмент)	2	4	50		AlterOS 7.5	ssh, Docker 20.10
QA - JSReports	2	4	50		AlterOS 7.5	ssh, Docker 20.10

Серверы системы должны быть объединены локальной вычислительной сетью с пропускной способностью не менее 100 Мбит/сек.

3 Сборка и развертывание интеграционных решений Enterprise Integrator

3.1 Сборка приложения Enterprise Integrator из исходных текстов

Переменные окружения, использование для сборки платформы:

- $work_dir – директория, где производится сборка;
- $PROJECT_GIT - абсолютный url корня проекта, хранения исходников потоков;
- JDK 1.8 и выше;
- В переменных окружения должна присутствовать коренная переменная JAVA_HOME;
- Установленный Integration Studio (загрузка архива со страницы https://github.com/wso2/product-ei/releases).

В операционной системе, где будет производиться сборка проектов, необходимо убедиться в наличии следующего окружения и при необходимости установить:

- MAVEN (https://maven.apache.org/install.html)
- JDK 1.8 и выше
- Git 2.17 и выше

Для сборки необходимо выполнить следующие шаги:

1. Загрузить проект из системы контроля версий:

```
cd $work_dir
git clone $PROJECT_GIT/eip_Сервисы
```

2. Открыть проект с использованием Integration Studio и перейти в меню для открытия проекта из каталога (см. Рисунок 1).

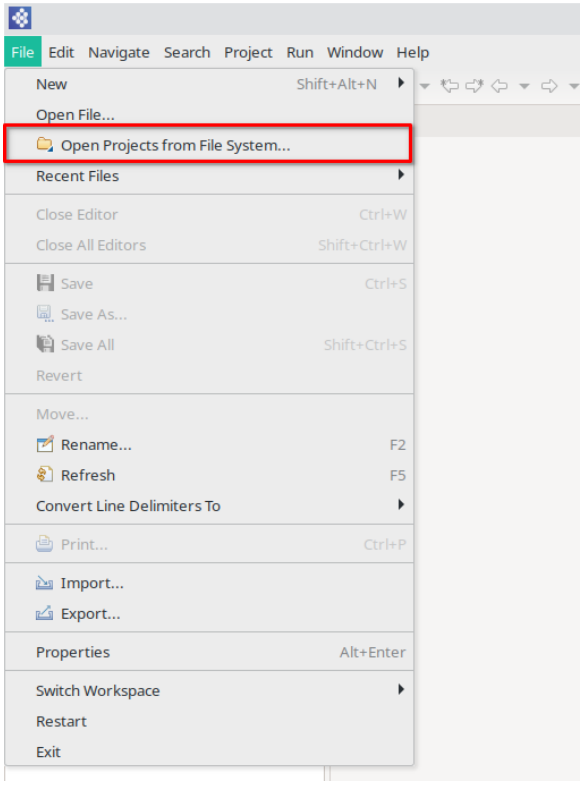


Рисунок 1. Открыть проект из каталога

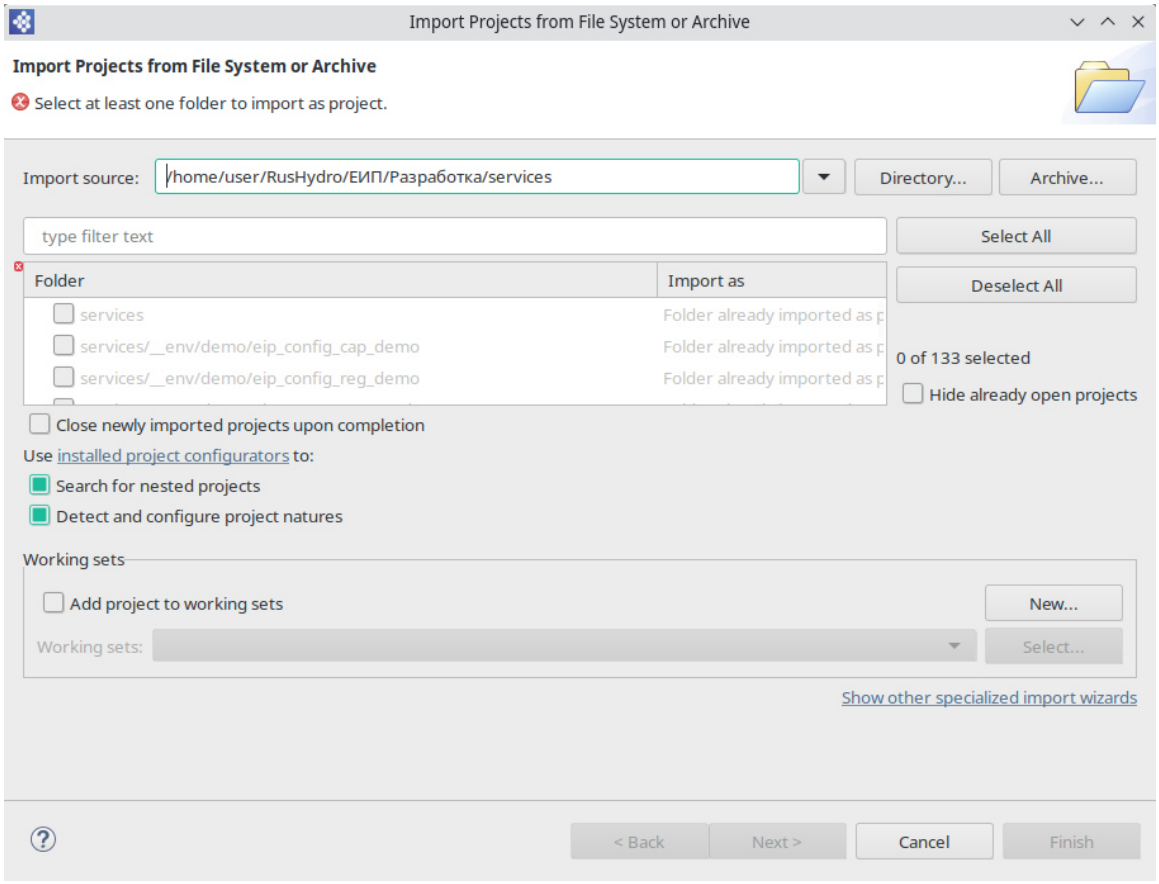
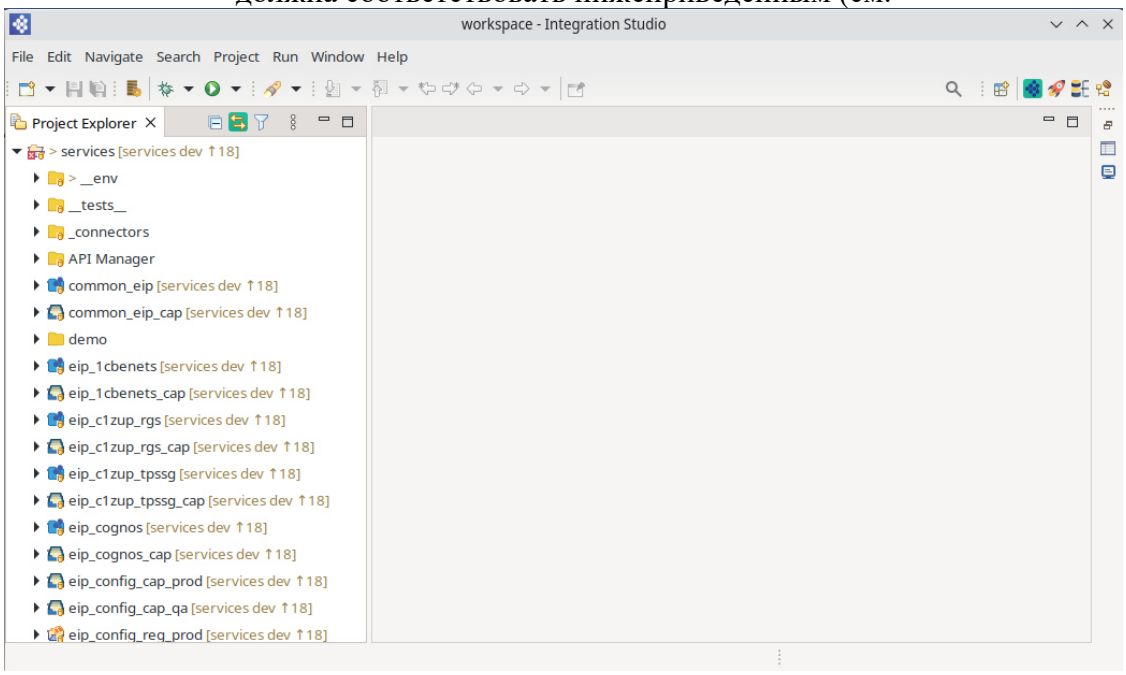


Рисунок 2. Анализ каталога Сервисы

3. Выберите каталог для проведения его анализа (автоматически) на предмет наличия проектов (см. Рисунок 2).

После импорта проектов структура проектов в обозревателе проектов должна соответствовать нижеприведенным (см.



4. Рисунок 3).

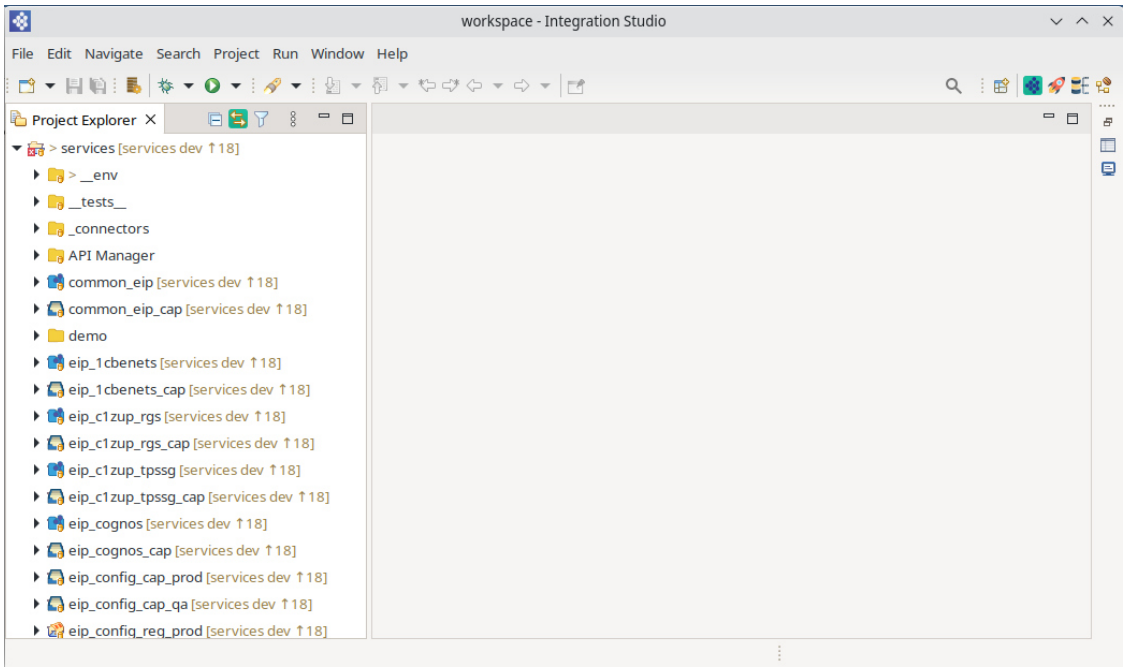


Рисунок 3. Древо проектов.

5. В дереве каталогов, последовательно выбирать проекты Composite Application Project (отображаемых пиктограммой [1]). В контекстном меню (нажать правой кнопкой мыши по проекту) и Выберите "Export Composite Application Project". (Для разных сред сформированы отдельные проекты Composite Application, с постфиксом PROD, QA, DEV).

6. Определите каталог, в который будет сохранен архив car (см. Рисунок 4).

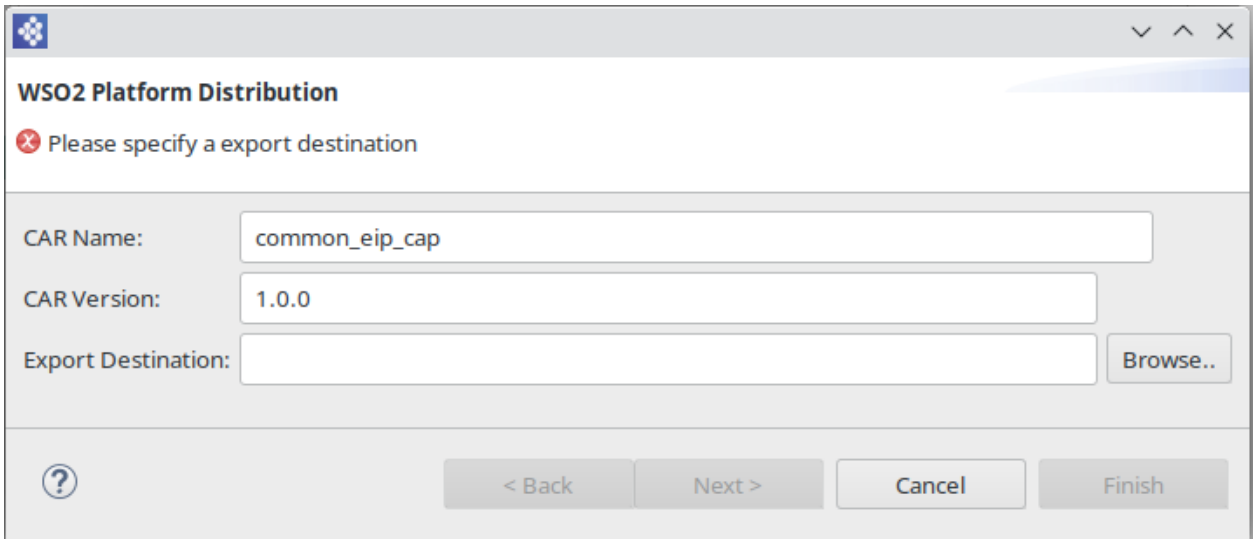


Рисунок 4. Формирование CAR файла.

Проекты, которые необходимо собрать, в зависимости от контура решения будут именоваться:

1 В зависимости от версии Integration Studio иконки могут отличаться

eip_<наименование_решения>_cap_<наименование_контура>

или

eip_<наименование_решения>_cap.

3.2 Конфигурирование DSS сервисов ESB профиля

Для конфигурирования DSS сервисов в части определения параметров подключения к источнику данных необходимо открыть проект сервисов DSS (eip_dss_qa для тестового контура; eip_dss_prod для продуктивного контура) в среде разработки Integration Studio и выбрать DSS сервис, который необходимо сконфигурировать (см. Рисунок 5).

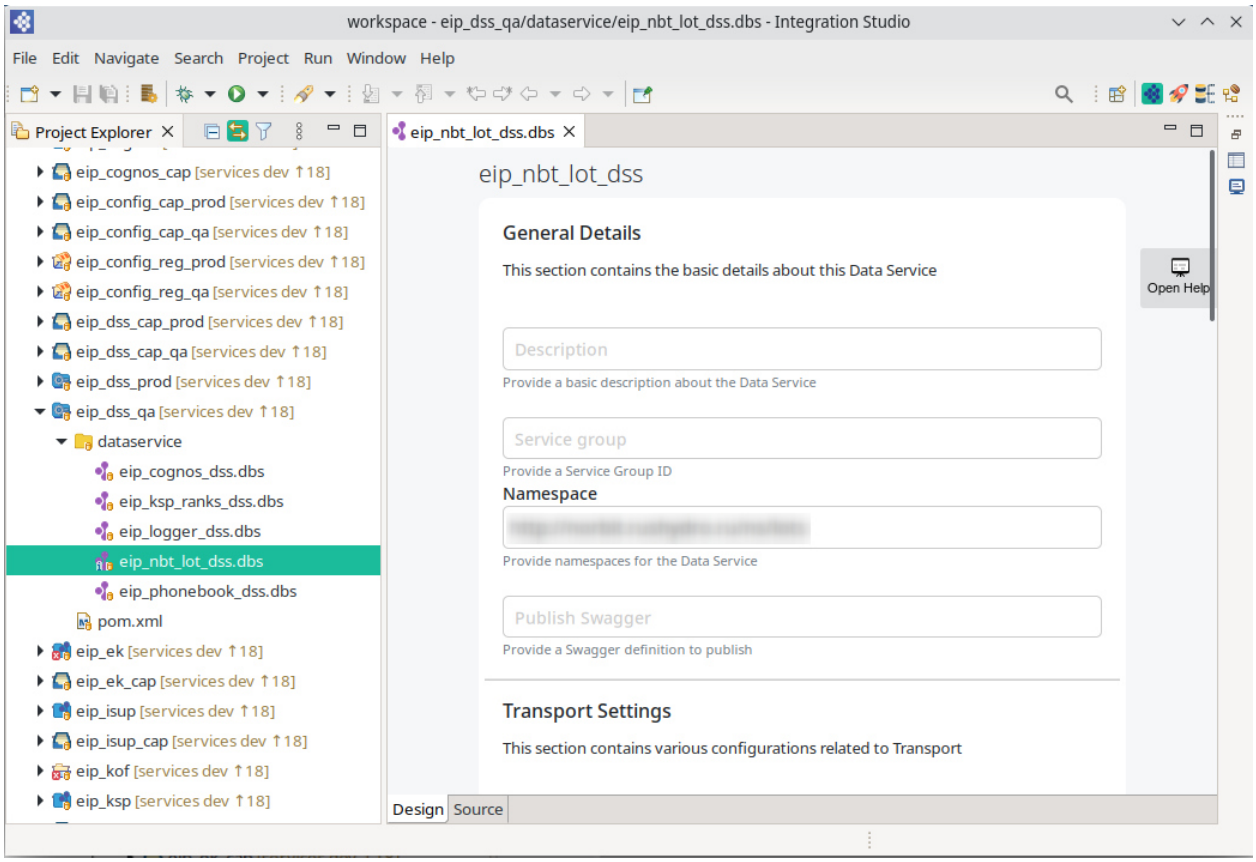


Рисунок 5 – Окно интерфейса конфигурирования DSS сервиса

В появившемся окне необходимо найти раздел «Data Sources» и, выбрав источник данных, который используется для выполнения запросов, нажать кнопку «Редактировать» (см. Рисунок 6).

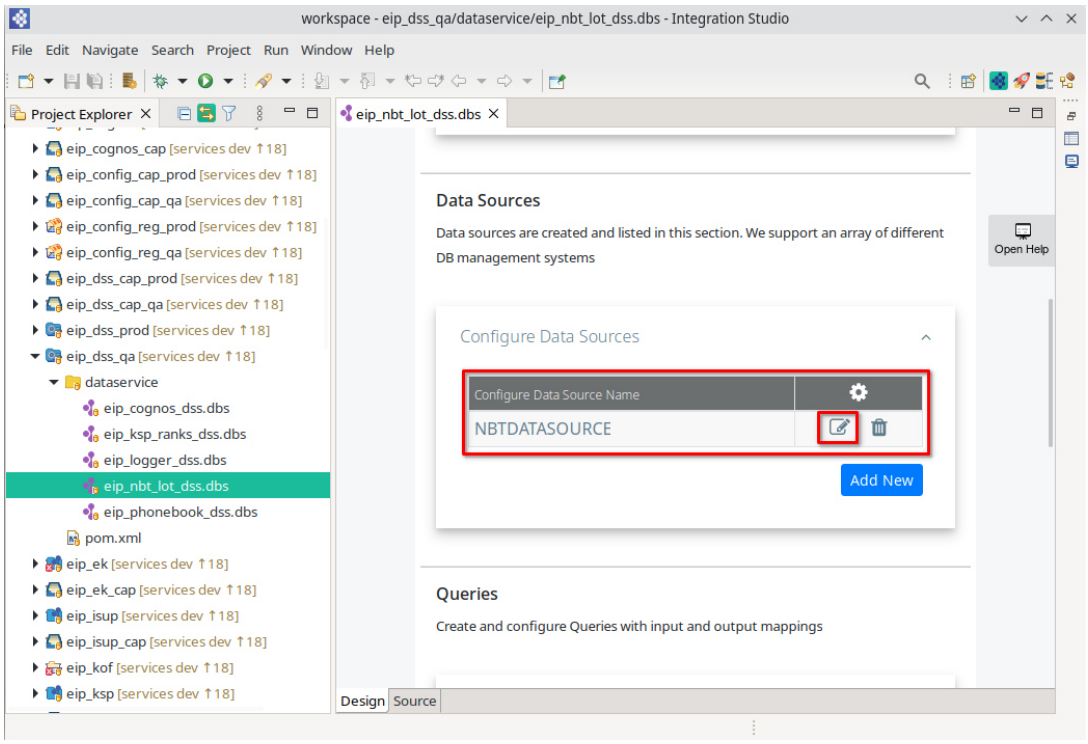


Рисунок 6 – Окно источников данных DSS сервиса.

В появившемся окне параметров источника данных DSS сервиса необходимо задать актуальные данные для подключения к источнику данных (см. Рисунок 7).

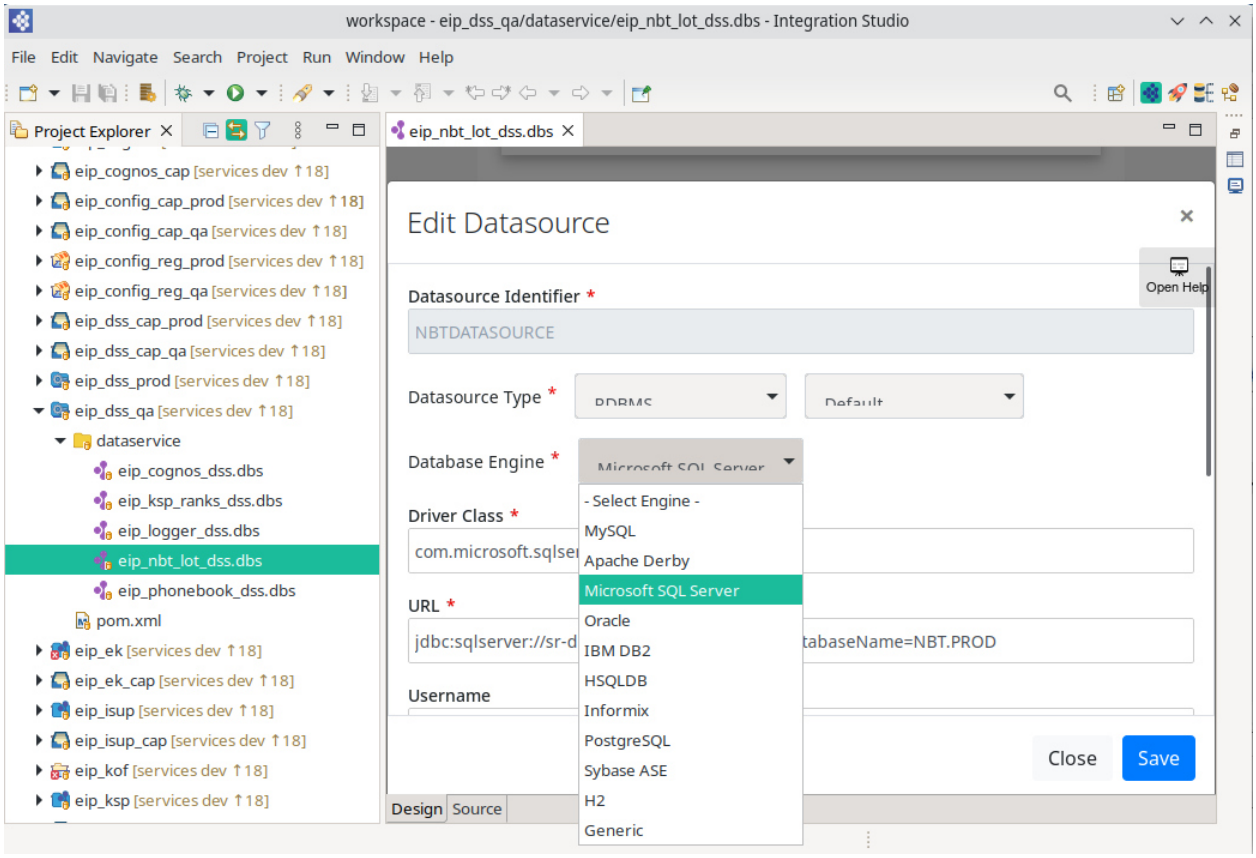


Рисунок 7 – Окно параметров источника данных DSS сервиса

После изменения параметров необходимо сохранить проект. После сохранения можно перейти к процессу установки артефакта, содержащего DSS сервисы на единицу развертывания. Для этого см. раздел 1.3, содержащий инструкцию по развертыванию.

3.3 Установка артефактов на серверы Enterprise Integrator

Разработанные сервисы необходимо скопировать в каталог развертывания <EI_HOME>/repository/deployment/server/carbonapps на каждый сервер.

Для удобства воспользуйтесь меню администратора, перейдите по ссылке (https://ei-domain-name:9443/carbon).

Выберите вкладку «Домашняя» > «Приложения Carbon> пиктограмма «Добавить».

В открывшемся интерфейсе системы выберите необходимый сформированный артефакт и нажать на кнопку «Загрузить» (см. Рисунок 8).

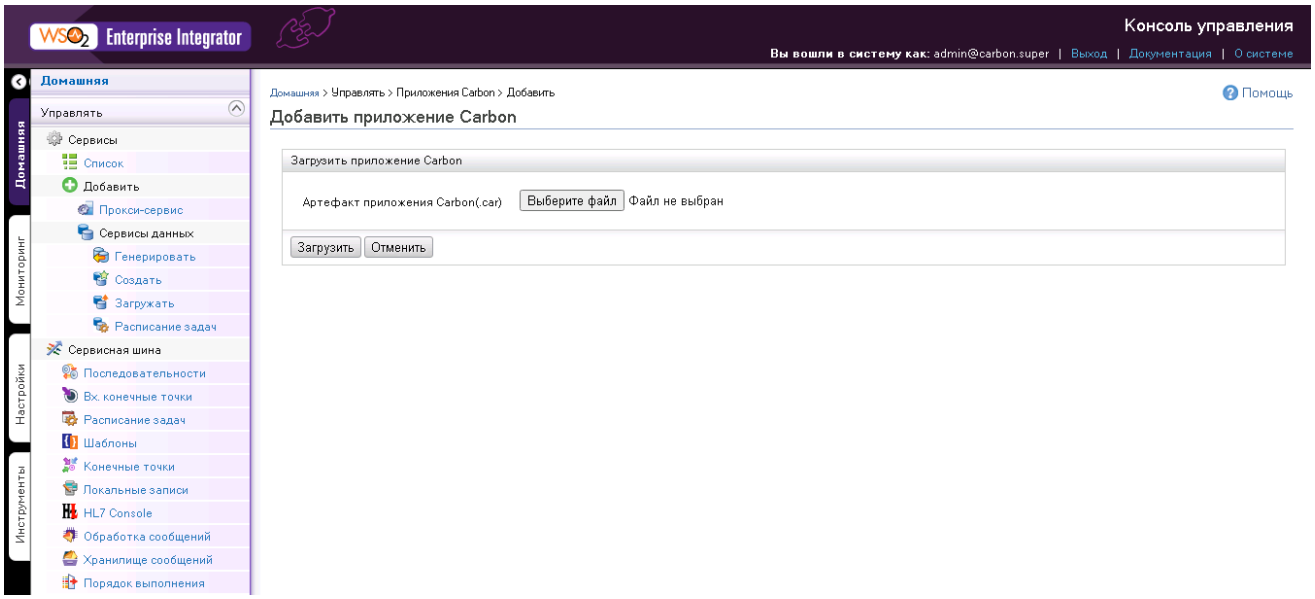


Рисунок 8. Графический интерфейс пункта «Добавить» (выбора) загрузка артефакта

4 Управление Carbon приложениями Enterprise Integrator

4.1 Описание Carbon

WSO2 Carbon – решение, которое предоставляет собой интегрированную многокомпонентную платформу. Позволяет быстро создавать, разворачивать, обновлять, удалять приложения и сервисы, созданные с помощью WSO2 Developer Studio. Реализована возможность устранять конфликты версий компонентов.

Базовая структура WSO2 Carbon – это единая система управления, с использованием графического интерфейса, для всех продуктов WSO2, таких как встроенный реестр, управление пользователями, транспорт, мониторинг, ведение журналов и кластеризация.

Согласно процедуре развертывания профиля ESB, “Инструкции по развертыванию кластера решений ЕИП”, “п 2.8 Создание скрипта запуска решения”, Carbon запускается автоматически, скриптом, при запуске операционной системы.

При успешном запуске Carbon, на ноде кластера, на которой развернуто решение, в журнале должна появиться запись вида:

```
INFO {org.wso2.carbon.core.internal.StartupFinalizerServiceComponent} - WSO2 Carbon started in 172 sec

INFO - CarbonUIServiceComponent Mgt Console URL  : https://ei-domain-name:9443/carbon/
```

Журнал в операционной системе располагается в каталоге < EI_HOME >/repository/logs, в фале wso2carbon.log, соответствующей ноды кластера, на которой запущен Carbon.

Для доступа к приложению к консоли управления Carbon откройте URL в браузере https://ei-domain-name:9433/carbon, в поле ввода учетных данных введите актуальные данные. Имена серверов необходимо подставлять согласно именам тестового или продуктивного контура. После успешной авторизации отобразится домашняя страница консоли управления.

Продукты WSO2 оперируют понятиями профиль (такими как профиль ESB (ESB profile), профиль аналитики (Analitics profile), профиль бизнес-процессов (Business Process profile) и т.д.).

Профиль – это набор шаблонов для проектирования интеграционной архитектуры организации через единую платформу ЕИП (WSO2EI).

Шаблон – это предустановленные конфигурации платформы для: шины данных (включая протоколы), консолидации и вывода данных, бизнес процессов.

4.1.1 Системные службы и пользовательский интерфейс

Особенностью продуктов WSO2 является разделение на интерфейс взаимодействия с пользователем и интерфейс взаимодействия служб:

- Взаимодействие с пользователем – осуществляется через консоль управления.

- Взаимодействие системных служб – часть решения, выполняющая основную работу (например, все сервисы исполняются именно в служебной части).

Эти части решения используют различные порты для взаимодействия. Выбор портов зависит от развернутых профилей. К примеру, служебная часть профиля ESB, в качестве транспорта, активно использует протоколы http и https на портах 8280 и 8243 соответственно. Это разделение активно используется для кластеризации: можно запустить несколько серверов приложений WSO2 на уровне взаимодействия служб для распределения нагрузки и только один сервер с консолью управления в кластере.

Взаимодействие с пользователем через консоль управления зависит от установленных профилей.

К примеру: консоль управления профиля ESB по умолчанию запускается на порту 9443, консоль управления профиля аналитики - 9643.

Пример прямой ссылки на консоль управления профилей ESB и аналитики продуктивного контура:

- https://ei-domain-name:9443/carbon
- https://ei-domain-name:9643/portal

Ниже будут описаны порты, по умолчанию используемые в решениях WSO2

В тестовом и продуктивном контурах, на каждой ноде кластера запущена консоль управления ESB профиля, что позволит посмотреть статистику каждого сервера и просмотреть мониторинг логов на каждом сервере кластера.

К примеру, вход в консоль на ноды продуктивного контура:

```
https://ei-domain-name:9443/carbon/admin/login.jsp
https://ei-domain-name:9443/carbon/admin/login.jsp
```

4.1.2 Смена базы данных

По умолчанию профиль ESB хранит все свои конфигурационные данные во встроенной базе H2. Согласно рекомендации разработчика WSO2, для продуктивных решений рекомендуется использовать промышленные СУБД. Продуктивное решение использует кластер, где в качестве СУБД используется PostgreSQL. Профили ESB и аналитики используют различные БД для хранения данных. В конфигурациях, при развертывании кластера указаны наименования БД используемые профилями ESB и аналитики (см. **«Инструкцию по развертыванию кластера решений ЕИП»).**

Для смены БД, профиля ESB, необходимо изменить файл <EI_HOME>/repository/conf/datasources/master-datasources.xml, поменяв в нем настройки для WSO2_CARBON_DB.

Для настройки БД профиля аналитики используется файл:

<EI_HOME>/wso2/analytics/conf/worker/deployment.yaml

Действие по настройке описано в **«Инструкция по развертыванию кластера решений ЕИП».**

При этом важно: *Разместить jar-файл с необходимым драйвером СУБД в <EI_HOME>/repository/components/lib. для профиля ESB и в <EI_HOME>/wso2/analytics/lib для профиля аналитики*.

4.1.3 Порты по умолчанию

Ниже перечислены порты по умолчанию, которые используются в WSO2 Enterprise Integrator (WSO2 EI), а также в ЕИП:

- Порты ESB:
  1. 9443 - транспорт сервлета HTTPS (URL-адрес консоли управления по умолчанию - https://ei-domain-name:9443/carbon).
  2. 8243 - Транспортный HTTPS.
  3. 8280 - Транспортный HTTP.
- Порты EI-Analytics:
  1. 9643 - транспорт сервлета HTTPS (URL-адрес консоли управления по умолчанию - https://ei-domain-name:9643/portal).
  2. 9444 - Используется для консоли управления EI-Analytics.
- 9445 - транспорт сервлетов HTTPS для среды выполнения EI-Business Process (URL-адрес консоли управления по умолчанию -https://ei-domain-name:9445/carbon).
- Порты сервера LDAP:
  1. 10389 - Используется в продуктах WSO2, которые предоставляют встроенный сервер LDAP.
- Порты KDC:
  1. 8000 - Используется для предоставления доступа к серверу ключей Kerberos.
- Порты мониторинга JMX Платформа WSO2 Carbon использует TCP-порты для мониторинга работающего экземпляра Carbon с помощью клиента JMX, такого как JConsole. По умолчанию JMX включен во всех продуктах.
  1. 11111 - RMIRegistry порт. Используется для удаленного мониторинга Carbon.
  2. 9999 - Порт RMIServer. Используется вместе с портом RMIRegistry, когда Carbon контролируется клиентом JMX, который находится за брандмауэром.
- Для кластеризации любого работающего экземпляра Carbon должен быть открыт один из следующих портов:
  1. 45564 - Открыт, многоадресная схема кластеризации.
  2. 4000 - Открыт, схема wka.
- Динамически создаваемые порты

1. Некоторые порты создаются динамически во время запуска сервера. Это связано с конкретными свойствами и конфигурациями, которые вступают в силу при запуске продукта. Идентификаторы этих портов будут меняться при каждом запуске сервера.

5 Консоль администрирования Enterprise Integrator

При входе в систему, на домашней странице отображается информация о среде размещения сервера, развернутой операционной системе, информации о вошедшем в систему пользователе, информации о конфигурации JVM и информации о реестре в пяти отдельных таблицах.

Carbon позволяет оперировать параметрами системы из консоли управления. Каждая вкладка консоли управления, отражает целевой набор параметров. Каждый из параметров позволяет просматривать, добавлять и настраивать сервисы, не прибегая к системе разработки Developer Studio.

В левой части интерфейса отражены четыре вкладки (см. Рисунок 9):

- **«Домашняя»**;
- «**Мониторинг**»;
- «**Настройки**»;
- «**Инструменты**».

Вкладка «**Домашняя**» содержит разделы, подразделы:

- «**Управление**»:
  - «**Сервисы**»;
  - «**Сервисная шина**»;
  - «**Коннекторы**»;
  - «**Инструменты учетных данных в артефактах WSO2**»;
  - «**Carbon приложения**»;
  - «**Модули**»;
  - «**Топики**»;
  - «**Выключение\перезапуск**».
- «**Реестр**»:
  - «**Просмотр**»;
  - «**Поиск**».

Вкладка «**Мониторинг**» содержит пункты:

- «**Журнал приложений**»;
- «**Системная статистика**»;
- «**Системный журнал**»;
- «**Поток сообщений**».

Вкладка «**Настройки**» содержит пункты:

- «**Пользователи и роли**»;
- «**Пользовательские истории**»;
- «**Хранилища ключей**»;
- «**Система логирования**»;

- «**Источники данных**»;
- «**Приемники событий**»;
- «**Роли сервера**»;
- «**Мультитентальность**» (Поддержка множества арендаторов, компаний - пользователей сервиса).

Вкладка «**Инструменты**» содержит пункты:

- «**WSDL2Java**» (средство генерации Java кода на основании WSDL описания сервиса);
- «**Java2WSDL**» (средство генерации WSDL на основании исходных кодов Java);
- «**Сделай это**» (средство для тестирования веб-сервисов сервис-ориентированных архитектур);
- «**WSDL валидатор**».

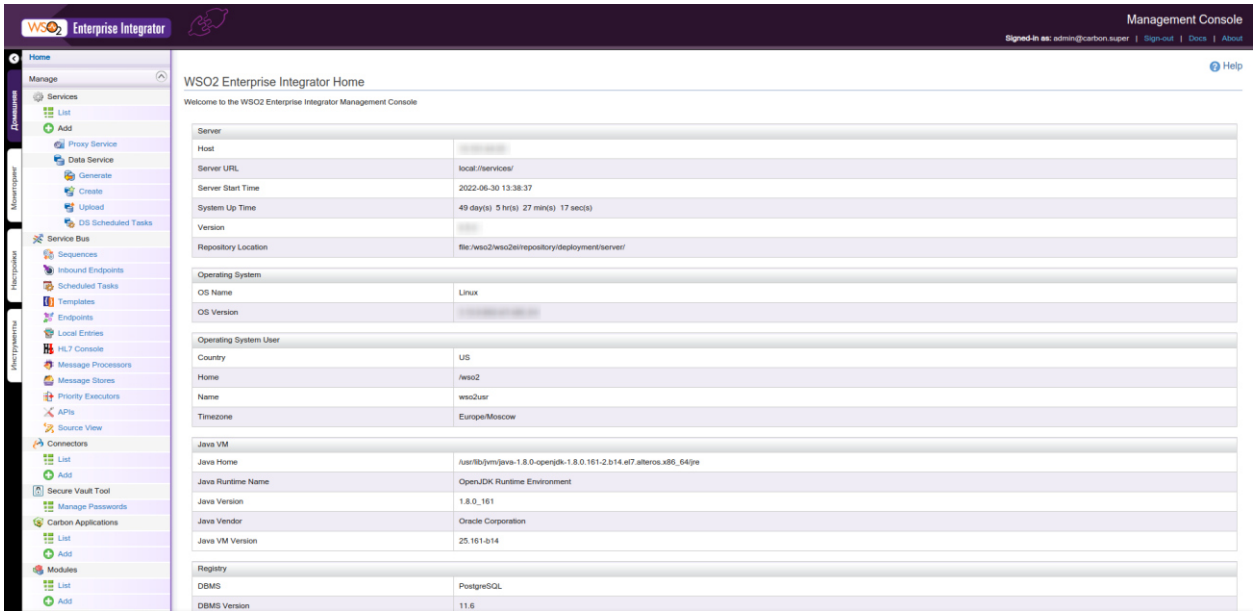


Рисунок 9. Графический интерфейс домашней страницы консоли управления, профиль ESB

5.1 Вкладка: «Домашняя» - «Управлять»

5.1.1 Раздел «Сервисы»

5.1.1.1 «Список»

Для перехода в интерфейс пункта «**Список**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Сервисы**» > «**Список**».

Графический интерфейс пункта «**Список**», отображает список, типы развернутых сервисов и элементы интерфейса, позволяющие управлять ими (см. Рисунок 10).

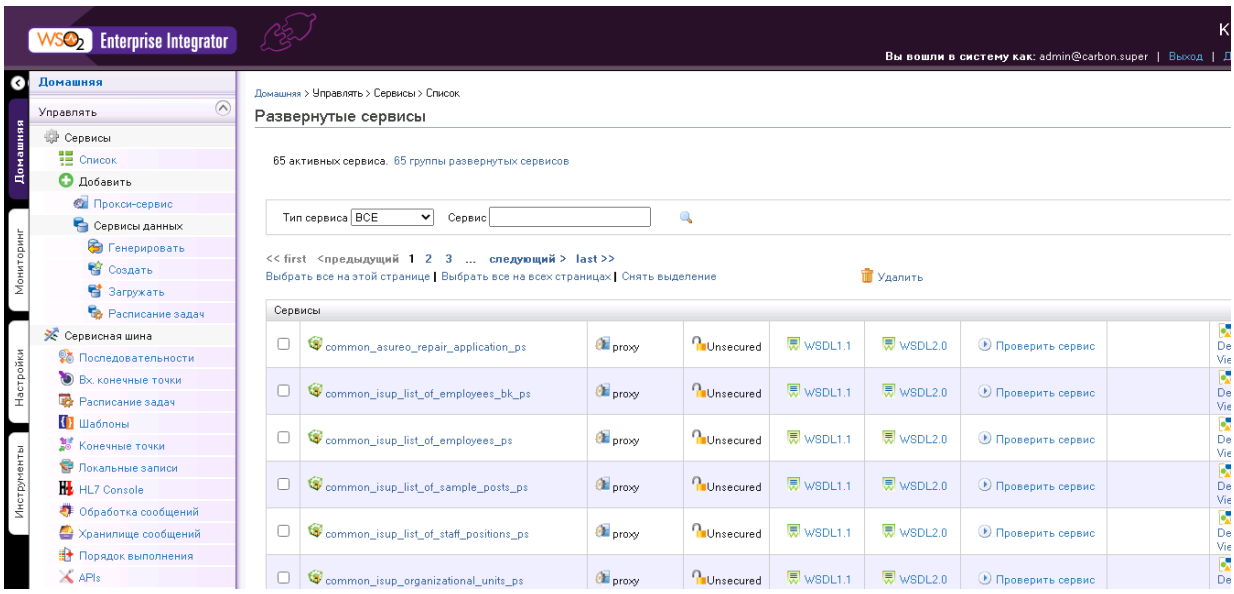


Рисунок 10. Графический интерфейс пункта «Список», отображающий список развернутых сервисов.

Имена сервисов и отображаемые параметры списка – это ссылки на внутренние утилиты платформы, которые позволяют перейти на сервисную панель или произвести различные действия (такие как: проверку, редактирование, тестирование, отключение) над выбранным сервисом (см. Рисунок 11).

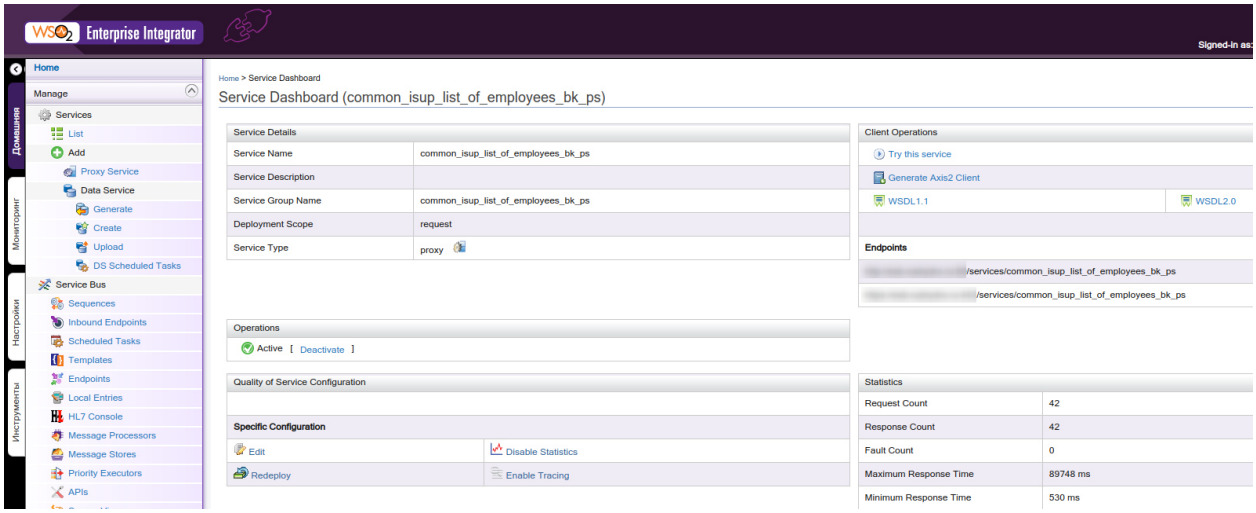


Рисунок 11. Графический интерфейс сервисной панели.

5.1.1.2 «Прокси сервис »

Прокси-сервис действует как сервис, размещенный в WSO2, и работает с существующим сервисом конечной точки («**endpoint**»). Прокси-сервис может быть создан и предоставлен для другого транспорта, схемы, WSDL и т.п., отличных от реального сервиса. Он может передавать сообщения до их доставки на конечную точку и ответы до того, как они достигнут клиента.

Для перехода в интерфейс пункта «**Прокси сервис**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Сервисы**» > «**Добавить**» > «**Прокси сервис**»

Графический интерфейс пункта «**Прокси сервис**», предоставляет перечень шаблонов, по критериям которых можно создать Прокси-сервис (см. Рисунок 8):

Transformer Proxy (Преобразующий прокси) - Создает прокси-сервис, который преобразует все входящие запросы с помощью XSLT, а затем перенаправляет их на заданные конечные точки. При необходимости ответы, возвращаемые из серверной службы, также могут быть преобразованы.

Защищенный прокси - Создает прокси с включенным WS-Security. Сервис будет обрабатывать WS-Security при поступлении запросов и перенаправлять их в незащищенную внутреннюю службу.

Прокси основанный на WSDL - Создает прокси-сервис из WSDL существующей веб-службы. Информация о конечной точке извлекается из удаленного хоста WSDL реальной службы.

Журналирующий прокси - Создает прокси-сервис, который регистрирует все входящие запросы и перенаправляет их на заданную конечную точку. При необходимости ответы, возвращаемые из серверной службы, могут быть записаны в журнал перед их отправкой клиенту.

Сквозной прокси - Создает простой прокси-сервис на указанной конечной точке. Прокси-сервис не выполняет обработку сообщений, проходящих через прокси-сервер.

Пользовательский прокси - Запустите мастер создания прокси-сервиса и создайте новый прокси-сервис с нуля. Настройте каждый аспект прокси-сервера, включая последовательности, конечные точки, транспортные протоколы и другие QoS параметры.

Если сервис успешно создан, он появится в поле списка «Сервисы» в интерфейсе пункта «**Список**» (см. Рисунок 12).

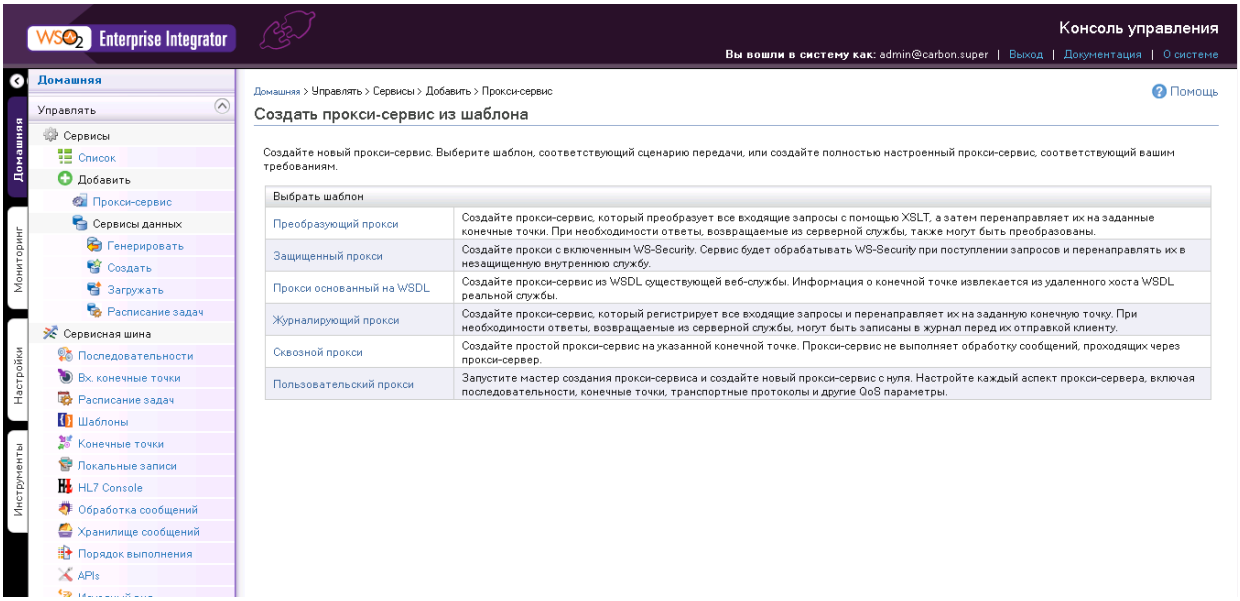


Рисунок 12. Графический интерфейс пункта «Прокси сервис», отображающий список шаблонов создания прокси-сервисов.

5.1.1.3 «Сервисы данных»

WSO2 DSS (Data Сервисы Server) - это веб сервер, предоставляющий интерфейс веб-службы для данных, хранящихся в таких источниках данных, как реляционные базы данных, файлы CSV, файлы Microsoft Excel и электронные таблицы Google.

Для перехода в пункт «**Сервисы данных**» выполните действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Сервисы**» > «**Добавить**» > «**Сервисы данных**»

Консоль управления позволяет создать сервис данных, загрузить предварительно сформированный в Developer Studio файл dbs – databse service, с настройками соединения, а также сформировать задачу в планировщике DSS.

Для генерации сервиса данных с предустановками, выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Сервисы**» > «**Добавить**» > «**Сервисы данных**» > «**Генерерировать**»

Система отобразит интерфейс с перечнем шагов по генерации сервиса данных (см. Рисунок 13).

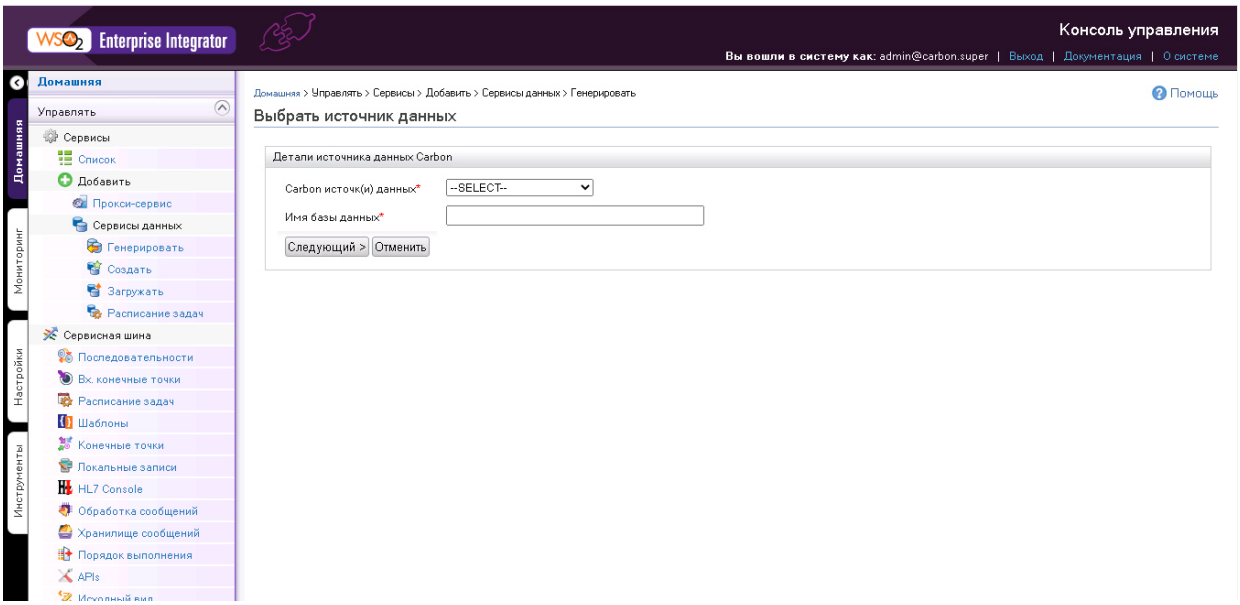


Рисунок 13. Графический интерфейс пункта «Generate».

Для генерации сервиса данных без предустановок, выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Сервисы**» > «**Добавить**» > «**Сервисы данных**» > «**Создать**»

Система отобразит интерфейс с перечнем шагов по созданию сервиса данных (см. Рисунок 14).

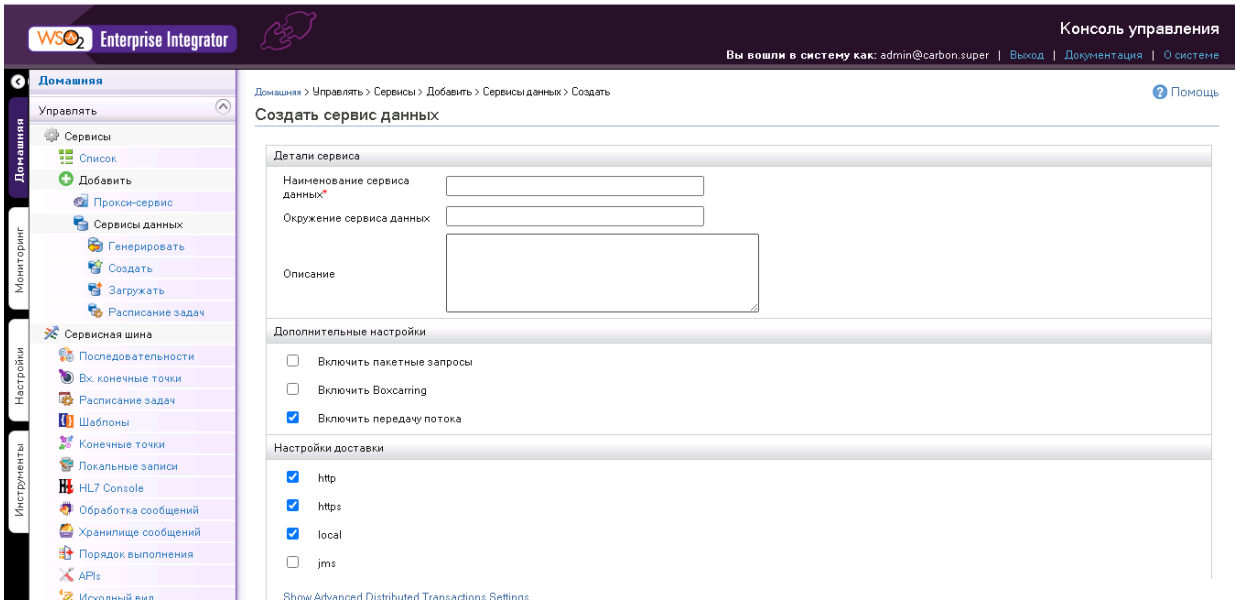


Рисунок 14. Графический интерфейс пункта «Create».

Для загрузки сервиса данных, выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Сервисы**» > «**Добавить**» > «**Сервисы данных**» > «**Загрузить**»

Система отобразит графический интерфейс с набором элементов, для загрузки файла сервиса данных (.dbs) (см. Рисунок 15).

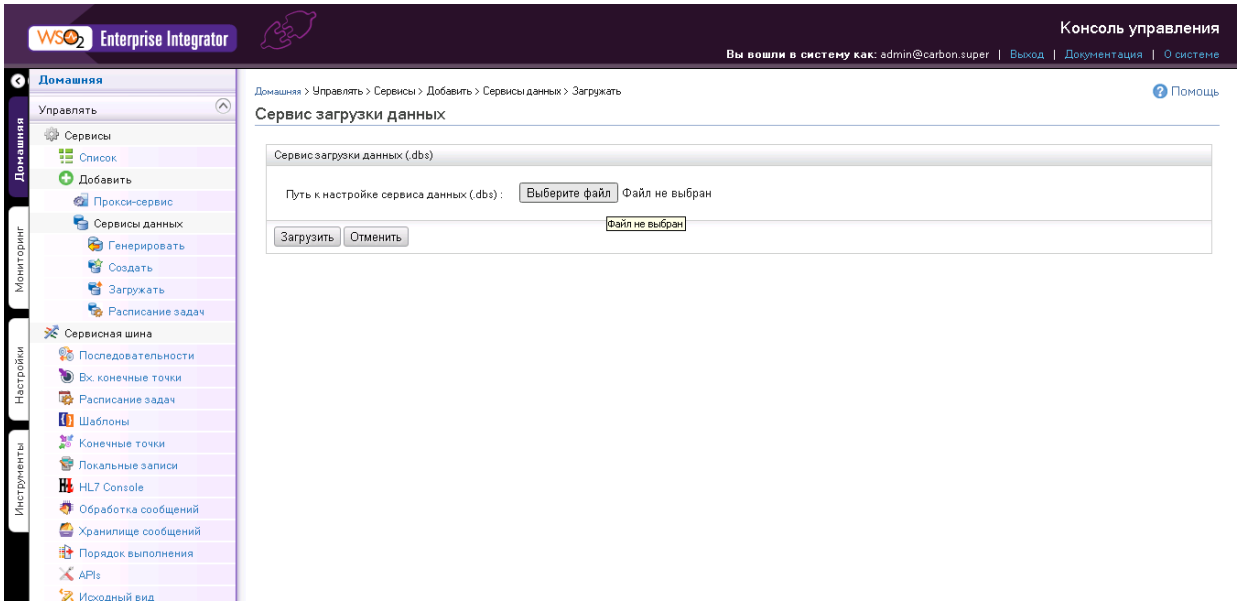


Рисунок 5. Графический интерфейс пункта «Загрузить».

Используя консоль управления WSO2 EI, можно планировать выполнение задач с определенной периодичностью. В процессе выполнения запланированной задачи, таймер инициализирует выполнение фрагмент кода, что позволяет выполнять запланированные задания с заданными интервалами. Задача - это пользовательский класс Java.

Развернув реализацию задачи в профиле ESB (см. раздел Написание задач), вы можете использовать консоль управления WSO2 EI для добавления, редактирования и удаления запланированных задач. Для добавления и планирования задач можно использовать конфигурацию пользовательского интерфейса или конфигурацию XML.

Таймер инициализирует выполнение фрагмента кода, что позволяет выполнять запланированные задания с заданными интервалами.

Атрибуты «count» и «interval» позволят планировать задачи на выполнение указанное количество раз с заданным интервалом.

Атрибута «once» позволит запланировать задачи на однократное выполнение.

Для добавления плановых заданий, выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Сервисы**» > «**Добавить**» > «**Сервисы данных**» > «**Расписание задач**»

Система отобразит графический интерфейс, позволяющий запланировать выполнение определенной задачи с указанной периодичностью (см. Рисунок 16).

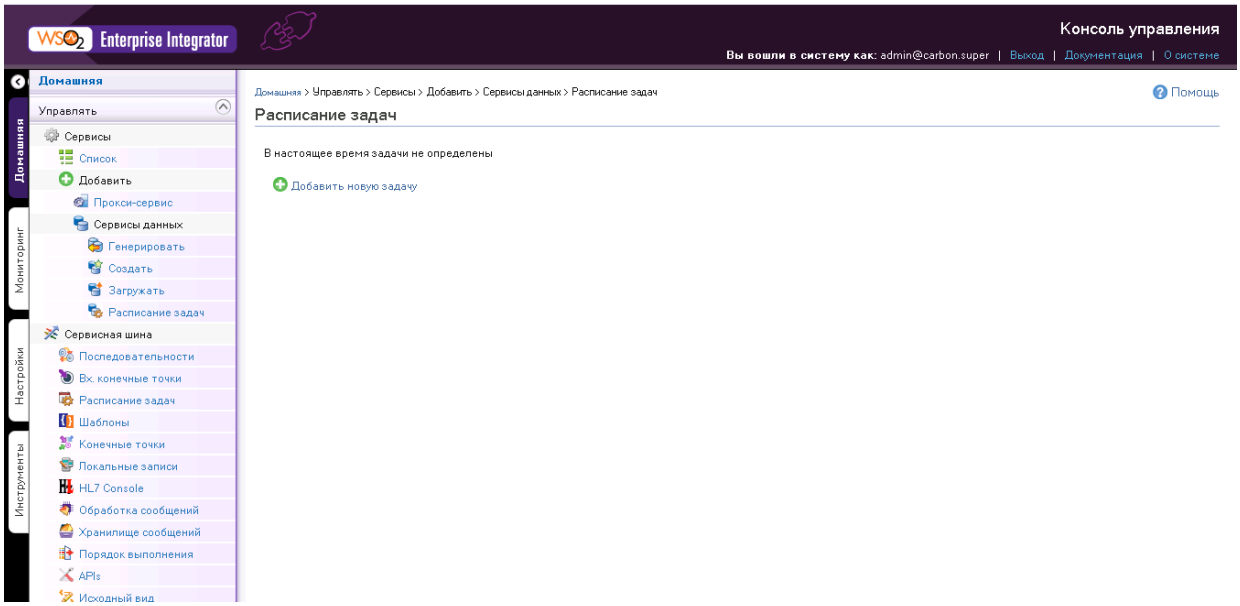


Рисунок 16. Графический интерфейс пункта «Расписание задач».

Обработка задач в кластерной среде.

В кластерной среде задачи по умолчанию распределяются между узлами сервера в соответствии с циклическим методом. При необходимости можно изменить это поведение обработки задач таким образом, чтобы задачи распределялись случайным образом или в соответствии с определенным правилом. Это параметр уровня сервера, который настраивается в файле: <EI_HOME>/conf/etc/tasks-config.xml.

Функционал системы позволяет настроить поведение обработки задач, указав в параметрах закрепленные серверы для выбранных задач. Этот параметр переопределяет конфигурацию на уровне сервера.

Запланированная задача будет выполняться только на одном из узлов (в данный момент времени) в кластерной среде. Задача перейдет на другой узел только в том случае, если первый узел выйдет из строя.

5.1.2 Раздел «Сервисная шина»

5.1.2.1 «Последовательности»

Sequence является именованной последовательностью медиаторов. Sequence применяется для того, чтобы избежать повторного написания уже определенной последовательности медиации (направления) сообщений. Система позволяет выбрать заранее определенную последовательность из реестра в качестве последовательности ввода/вывода/сбоя для прокси-службы без добавления каких-либо встроенных конфигураций.

Можно создать последовательность в ESB Config project или в реестре, а затем добавить ее в медиатор информационного потока или обратиться к ней из медиатора последовательности в том же ESB Config project или к другому проекту в этой рабочей области Eclipse.

WSO2 Средства ESB позволяют создать проект ресурсов реестра, который можно использовать для хранения ресурсов и коллекций, которые требуется развернуть в реестре Carbon Server через проект композитного приложения (C-App – Carbon Application). При создании последовательности ее можно сохранить как динамическую последовательность в проекте "Ресурс реестра" и обратиться к этой последовательности из промежуточного потока. Во время выполнения при развертывании CAR-файла как с помощью проекта Registry Resource, так и с помощью промежуточного потока WSO2 ESB выполняет поиск и использует последовательность из реестра.

Для перехода в интерфейс пункта «**Последовательности**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Сервисная шина**» > «**Последовательности**»

Система отобразит графический интерфейс, на котором расположены две вкладки отражающие существующие последовательности и элементы интерфейса для управления ими (см. Рисунок 17).

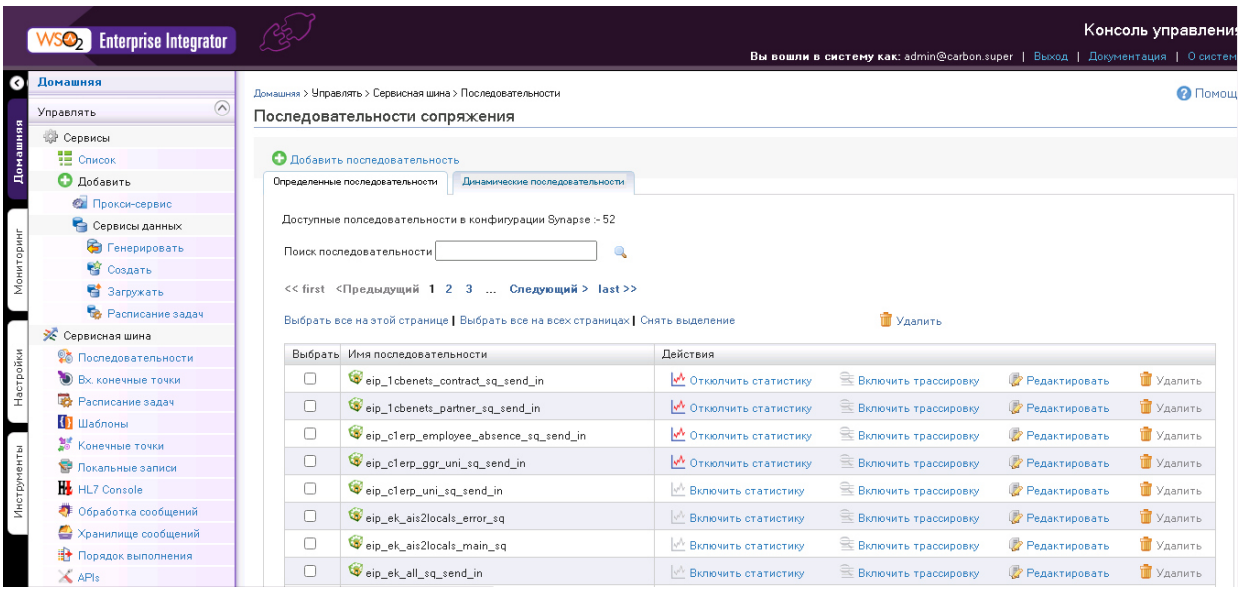


Рисунок 17. Графический интерфейс пункта «Последовательности».

Вкладка «Определенные последовательности» отображает последовательности, сохраненные в конфигурации Synapse.

Вкладка «Динамические последовательности» отображает последовательности, сохраненные в реестре.

Для добавления новой последовательности, нажмите кнопку «Добавить Последовательности».

Для удаления последовательности установите влаг выбора напротив этой последовательности и нажмите пиктограмму «Delete».

Для редактирования последовательности нажмите пиктограмму «Редактировать», соответствующий необходимой последовательности.

5.1.2.2 «Входящая конечная точка»

Входящая конечная точка - это точка входа сообщения, которая может вводить сообщения непосредственно с транспортного уровня на уровень передачи без прохождения через механизм Axis2, то есть могут быть определены полностью динамически (конфигурация читается в RunTime). В зависимости от протокола, поведение входящей конечной точки может быть прослушивающим, опрашивающим или основанным на событиях. Основная суть Inbound Endpoint – полностью динамическое определение.

Опрашивающая входящая конечная точка способен опрашивать источники сообщений, которые работают по специальным протоколам (такие как JMS, File) с определенным интервалом. Полученные сообщения будут переданы в определенный sequence для дальнейшей обработки. Задачи получения данных реализованы во внутренних механизмах WSO2 EI, поэтому могут быть сконфигурированы как интервал получения данных, так и механизмы координации выполнения обработки полученных сообщений. Такие точки являются асинхронными.

Прослушивающая входящая конечная точка открывает интерфейс для получения сообщений от внешних клиентов, которые могут отправить сообщения в этот интерфейс. Примером такой точки может быть HTTP входящая конечная точка , который динамически создает HTTP интерфейс для получения запросов от внешних клиентов. Прослушивающие входящие конечные точки поддерживают двухсторонние операции и являются синхронными.

Входящая конечная точка на основе событий опрашивает только один раз, чтобы установить соединение с удаленным сервером.

Для перехода в интерфейс пункта «**Входящая конечная точка**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Сервисная шина**» > «**Входящие конечные точки**»

Система отобразит графический интерфейс с набором элементов, которые позволяют создавать и редактировать входящие конечные точки. (см. Рисунок 18).

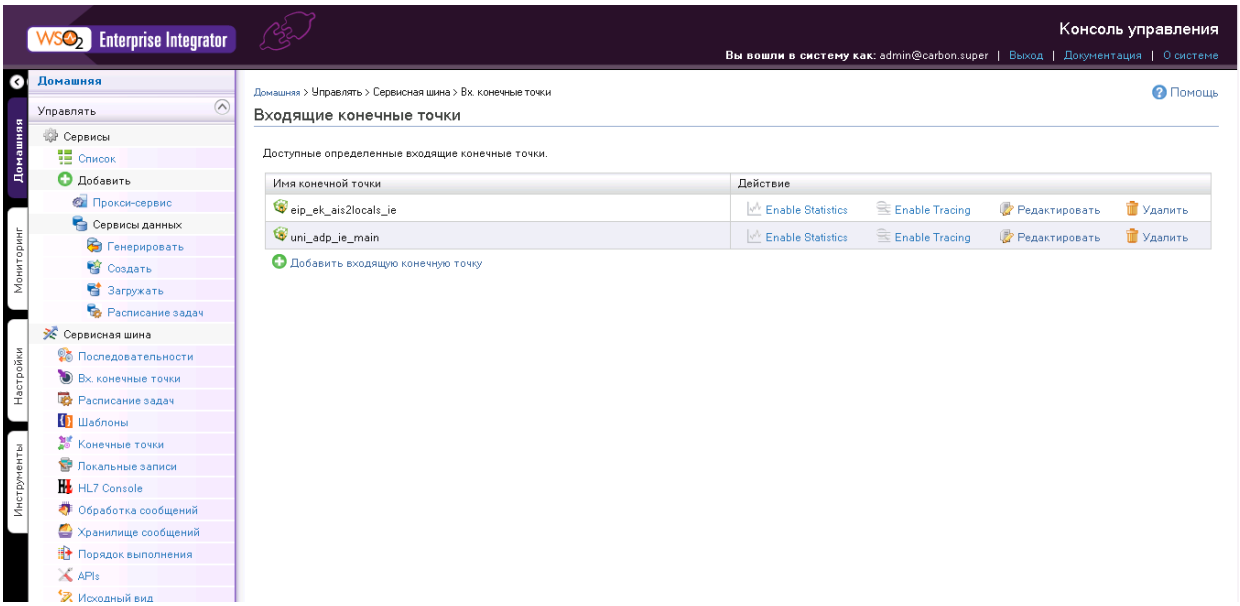


Рисунок 18. Графический интерфейс пункта «Входящие конечные точки».

На рисунке 19 представлена архитектура входящей конечной точки.

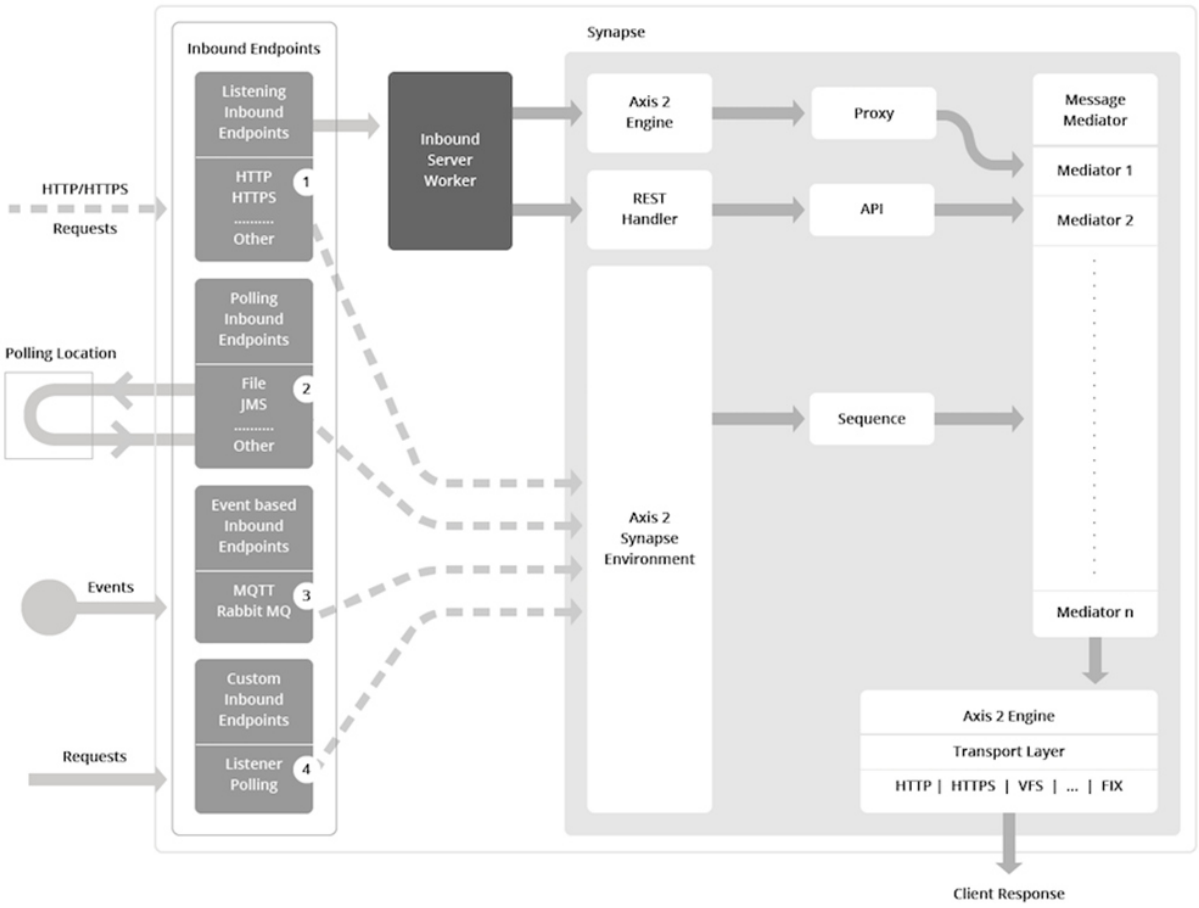


Рисунок 19. Архитектура входящей конечной точки.

5.1.2.3 «Расписание задач»

Задача - это пользовательский класс Java. Задачи могут быть настроены, как на периодическое выполнение, так и на выполнение по Cron выражению.

Задача предварительно должна быть скомпилирована и упакована в файл с расширением (.JAR). JAR-файл должен быть размещен <WSO2_HOME>/repository/components/lib.

Для перехода в интерфейс пункта «**Расписание задач**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Сервисная шина**» > «**Расписание задач**»

Система отобразит графический интерфейс с набором элементов, которые позволяют создавать и редактировать расписания задач. (см. Рисунок 20).

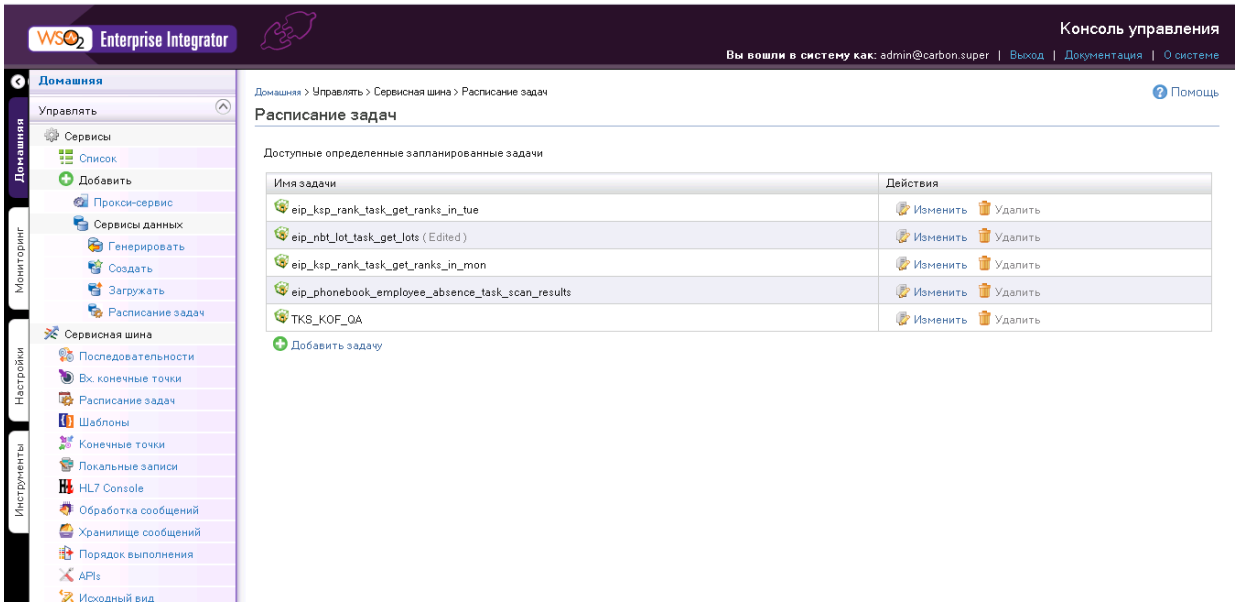


Рисунок 20. Графический интерфейс пункта «Расписание задач».

5.1.2.4 «Шаблоны»

Шаблоны позволяют упростить управление системой и минимизировать дублирование конфигураций, создавая прототипы, которые пользователи могут использовать столько, сколько нужно для выполнения поставленных задач.

Функции системы позволяют создавать шаблоны двух типов:

- Шаблон конечной точки;
- Шаблон последовательности.

Для перехода в интерфейс пункта «**Шаблоны**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Сервисная шина**» > «**Шаблоны**»

Система отобразит графический интерфейс с набором элементов, которые позволяют создавать и редактировать шаблоны (см. Рисунок 21).

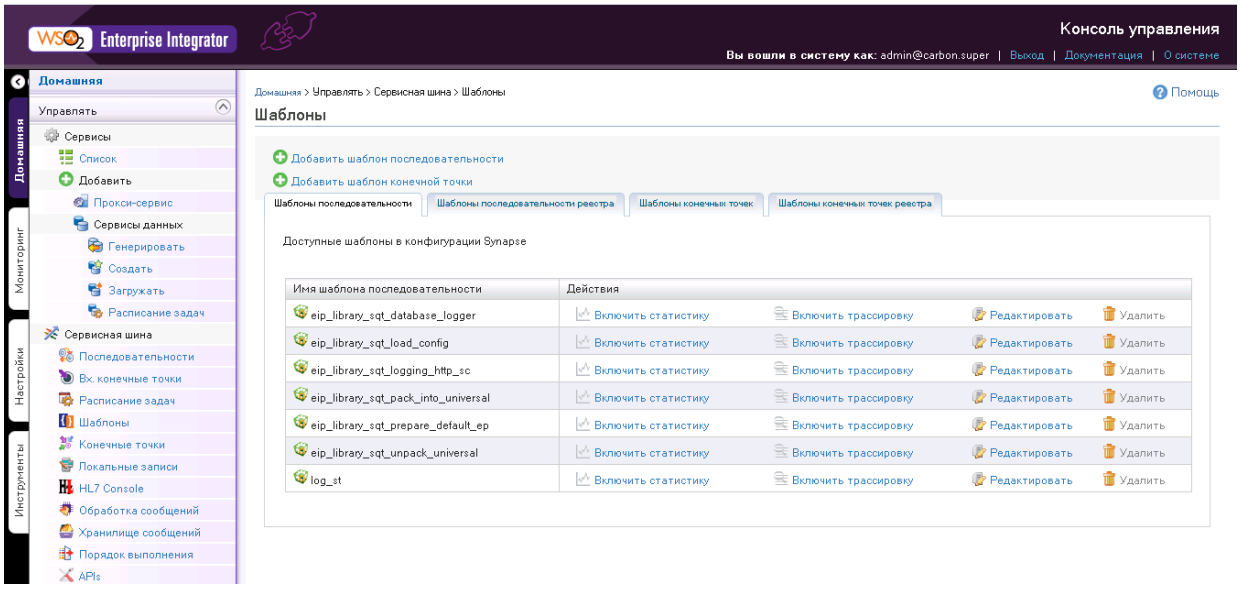


Рисунок 21. Графический интерфейс пункта «Шаблоны».

Шаблон конечной точки - это обобщенная форма настройки конечной точки, используемая в ESB. В отличие от шаблона последовательностей, шаблоны конечных точек всегда параметризуются с помощью префиксного значения «$» (не выражения XPath). Шаблон конечной точки - артефакт, который превращает шаблон конечной точки в конкретную конечную точку. Другими словами, шаблон конечной точки был бы бесполезен без конечной точки шаблона, ссылающейся на него. Это семантически похоже на связь между шаблоном последовательности и посредником шаблона вызова.

Шаблон последовательности - это параметризованная последовательность, обеспечивающая абстрактную или обобщенную форму последовательности, определенной в ESB. Параметры шаблона определяются в виде оператора XPath вызывающего шаблон и заполняя параметры статическими значениями выражений XPath с помощью посредника шаблона вызова, который превращает шаблон последовательности в конкретную последовательность.

5.1.2.5 «Конечные точки»

Конечная точка определяет внешнее назначение для исходящего сообщения через WSO2 EI. Как правило, конечная точка - это адрес прокси-службы, которая выступает в качестве внешнего интерфейса для реальной службы.

Для перехода в интерфейс пункта «**Конечные точки**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Сервисная шина**» > «**Конечные точки**»

Система отобразит графический интерфейс с набором элементов, которые позволяют создавать и редактировать конечные точки (endpoints) (см. Рисунок 22).

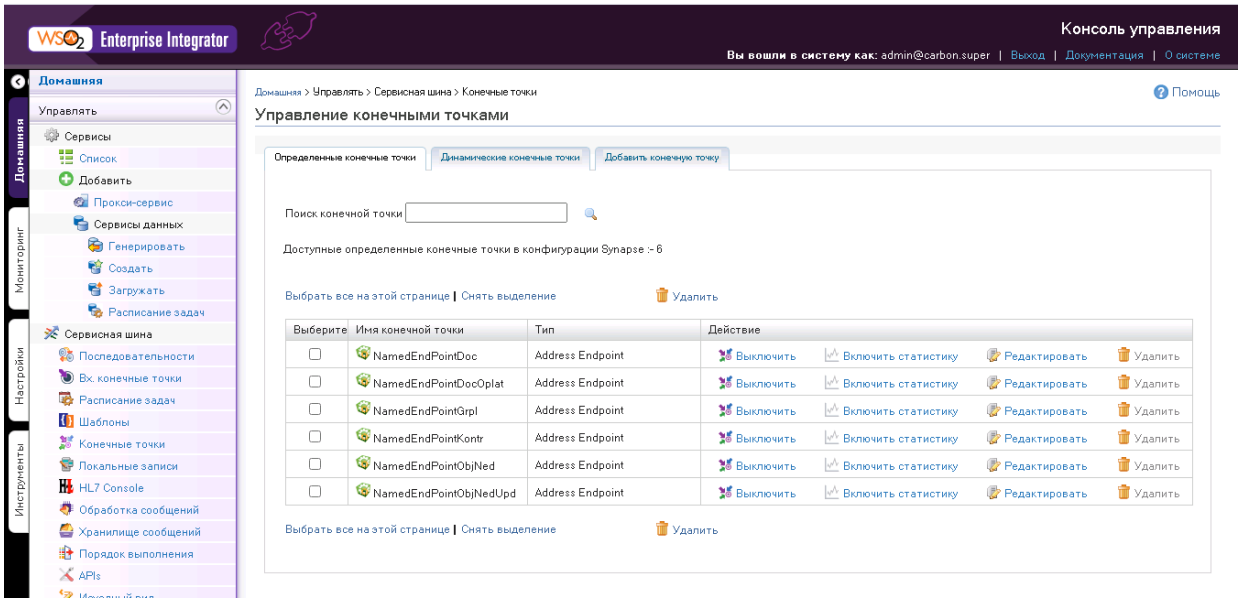


Рисунок 22. Графический интерфейс пункта «Конечные точки».

Вкладка «Динамические конечные точки » - позволяет редактировать и удалять конечные точки, сохраненные в реестре.

Вкладка «Добавить Конечные точки» - отражает список возможных типов конечных точек, которые можно создать в консоли управления.

5.1.2.6 «Локальные записи»

Локальный реестр может использоваться для хранения конфигураций и метаданных, специфичных для отдельно взятого узла ESB. Локальный реестр не может быть общим для всех узлов ESB в кластере или одноранговых узлках ESB.

Путь подключения локального реестра: - /_system/local

Локальные записи реестра хранят строки/текст, URL-адреса, XSD / XSLT, для последующего взаимодействия с ними из медиаторов. Доступ к этим локальным записям можно получить через медиаторы / последовательности с помощью следующей записи:

```
<localEntry key="validate_schema_XSLT">
```

Для перехода в интерфейс пункта «**Локальные записи**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Сервисная шина**» > «**Локальные записи**»

Система отобразит графический интерфейс с набором элементов, которые позволяют создавать и редактировать локальные записи (см. Рисунок 23).

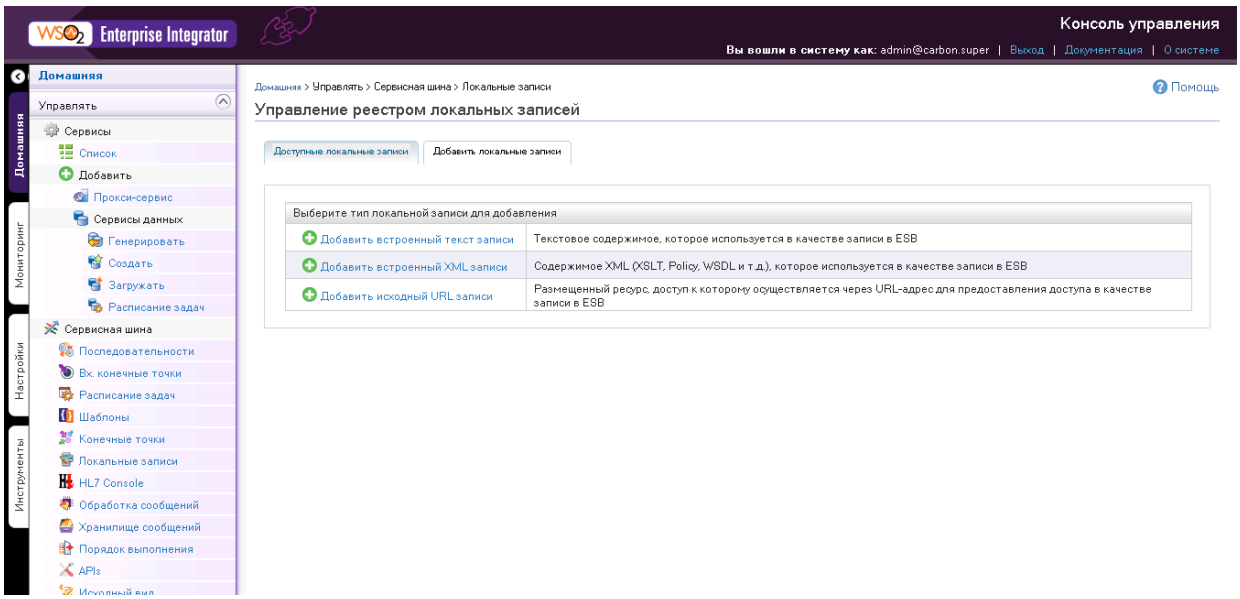


Рисунок 23. Графический интерфейс пункта «Локальные записи».

Локальные записи имеют три типа:

- Текстовая запись - текстовое содержимое, которое используется в качестве записи в ESB.
- XML запись - Содержимое XML (XSLT, Policy, WSDL и т.д.), которое используется в качестве записи в ESB.
- URL запись - Размещенный ресурс, доступ к которому осуществляется через URL-адрес для предоставления доступа в качестве записи в ESB.

5.1.2.7 «HL7 консоль »

Консоль управления HL7, позволяет взаимодействовать с транспортным протоколом по передаче сообщений определенного формата Health Level 7 International (HL7). HL7.

Для перехода в интерфейс пункта «**HL7 Console**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Сервисная шина**» > « **HL7 Консоль** »

Система отобразит графический интерфейс с набором элементов, которые позволяют создать и управлять прокси-службами используя транспорт HL7 (см. Рисунок 24).

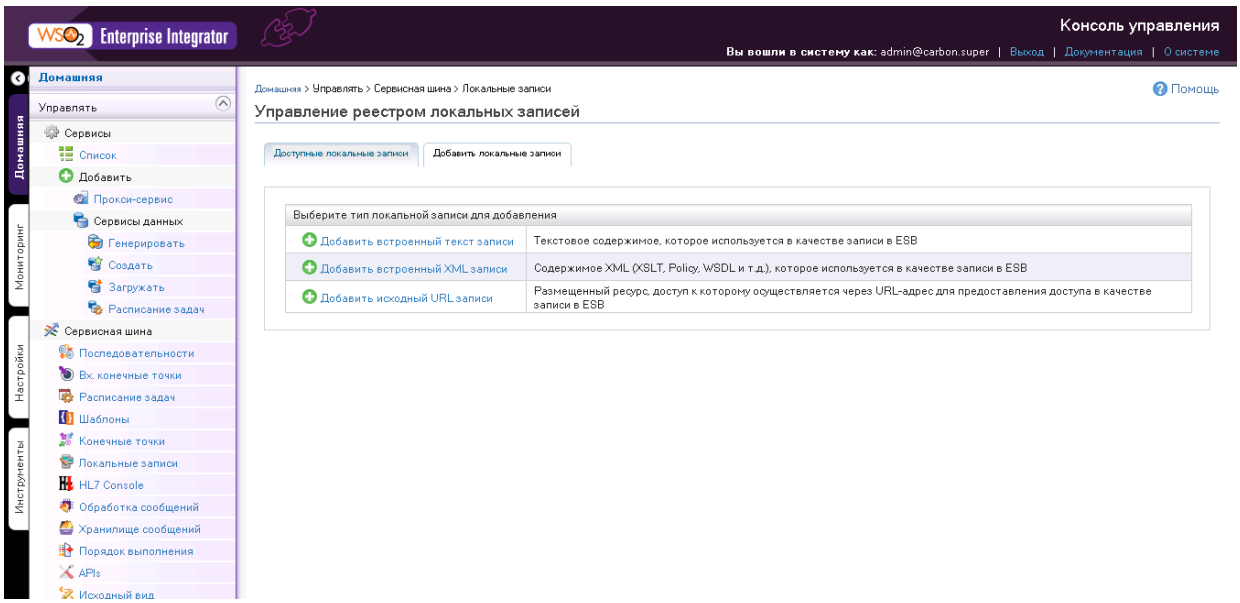


Рисунок 24. Графический интерфейс пункта «HL7 Консоль».

Для установки транспорта HL7, выполните следующие действия:

1. Выберите вкладку «**Конфигурация**» **->** «**Особенности**».

2. На вкладке доступные компоненты выберите репозиторий http://dist.wso2.org/p2/carbon/releases/turing. Если его нет в списке, нажмите кнопку «**Добавить репозиторий»** и добавьте его.

3. Тип **HL7** в поле Фильтр по имени объекта снимите флажок группировать объекты по категориям и нажмите кнопку «**Найти репозиторий»**.

4. Из списка доступных функций выберите и установите **Axis2 transport HL7**. Обязательно перезагрузите ESB после завершения установки.

5. Включите прослушивать и отправитель транспорт **HL7**.

WSO2 ESB поддерживает широкий спектр транспортов включая HTTP / s, JMS и VFS, а также доменные транспорты, такие как FIX. Все транспорты WSO2 прямо или косвенно основаны на платформе Apache Axis2 transports framework.

5.1.2.8 « Обработчик сообщений »

Обработчик сообщений используется для доставки сообщений, которые были временно сохранены в хранилище сообщений. Этот подход полезен для обслуживания трафика серверных служб, которые могут принимать сообщения только с заданной скоростью, в то время как входящий трафик в профиль ESB поступает с разной скоростью.

Для перехода в интерфейс пункта «**Message Processor**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Сервисная шина**» > «**Обработчик сообщений**»

Система отобразит графический интерфейс с набором элементов, которые позволяют создавать или редактировать существующие обработчики сообщений (см. Рисунок 25).

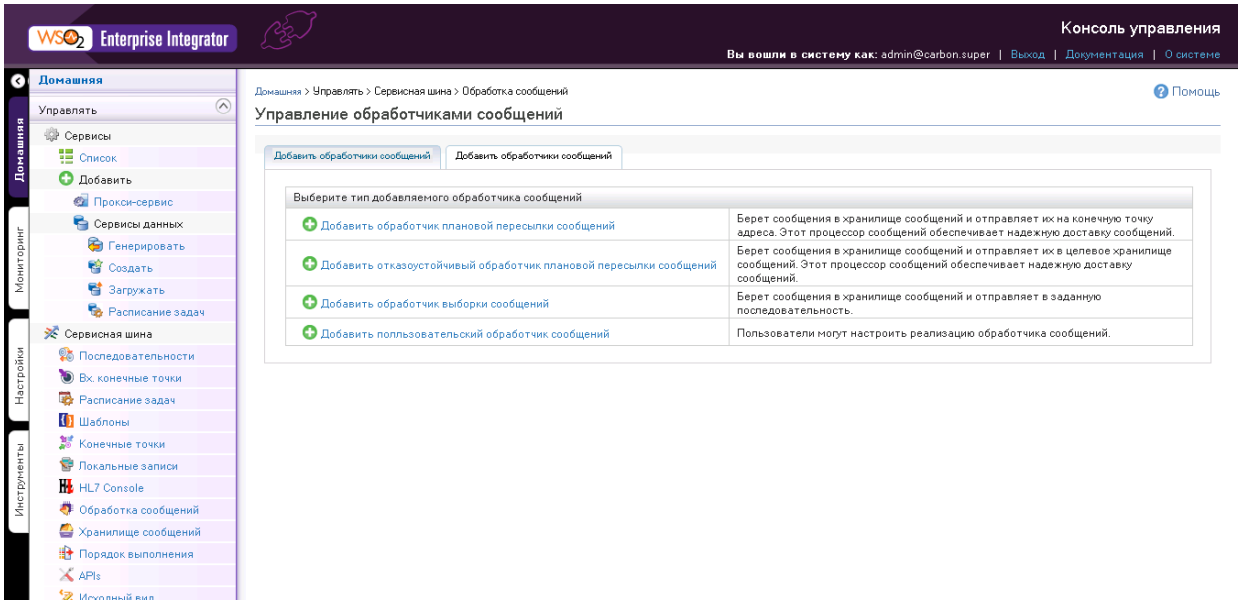


Рисунок 25. Графический интерфейс пункта «Обработчик сообщений».

Обработчики сообщений делятся на четыре типа:

Обработчик плановой пересылки сообщении - берет сообщения в хранилище сообщений и отправляет их на конечную точку адреса. Этот процессор сообщений обеспечивает надежную доставку сообщений.

Отказоустойчивый обработчик плановой пересылки сообщений - берет сообщения в хранилище сообщений и отправляет их в целевое хранилище сообщений. Этот обработчик обеспечивает надежную доставку сообщений.

Обработчик выборки сообщений берет сообщения в хранилище сообщений и отправляет в заданную последовательность.

Пользовательский обработчик сообщений пользовательская настройка реализации обработчика сообщений.

5.1.2.9 «Хранилище сообщений»

ESB профиль содержит список предустановленных шаблонов создания хранилищ сообщений. Используя хранилище медиаторов, для хранения сообщений в хранилище сообщений, используйте обработчик сообщений, для доставки сообщений из хранилища в серверную службу с заданной скоростью. Использование обработчика и хранилищ сообщений позволяет реализовать различные шаблоны обмена сообщениями и интеграции.

Несколько обработчиков сообщений могут использовать одно и то же хранилище сообщений. Например, в кластерной среде каждый из узлов может иметь экземпляр одного и того же обработчика сообщений, каждый из которых будет подключаться к одному и тому же хранилищу сообщений и равномерно получать сообщения из него (см. Рисунок 26).

Хранилище сообщений действует как менеджер и гарантирует, что сообщения обрабатываются только одним обработчиком сообщений, предотвращая дублирование сообщений. Существует возможность дополнительно контролировать, на каких узлах работает обработчик сообщений, указав закрепленные серверы.

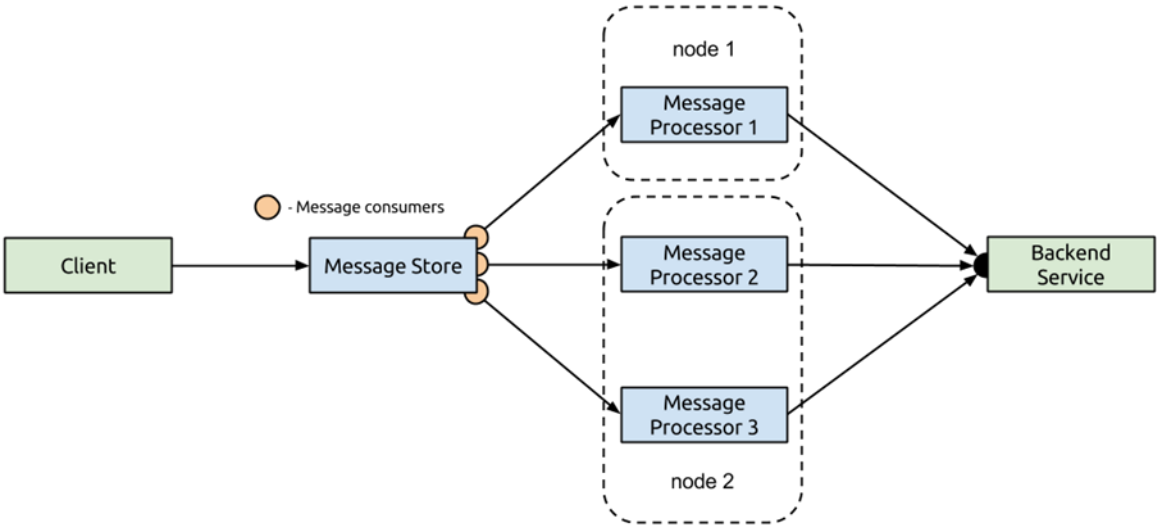


Рисунок 26. Схема взаимодействия Хранилище сообщений - обработчик сообщений.

Для перехода в интерфейс пункта «Хранилище сообщений», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Сервисная шина**» > «**Хранилище сообщений**»

Система отобразит графический интерфейс с набором элементов, которые позволяют создавать и редактировать хранилища сообщений (см. Рисунок 27).

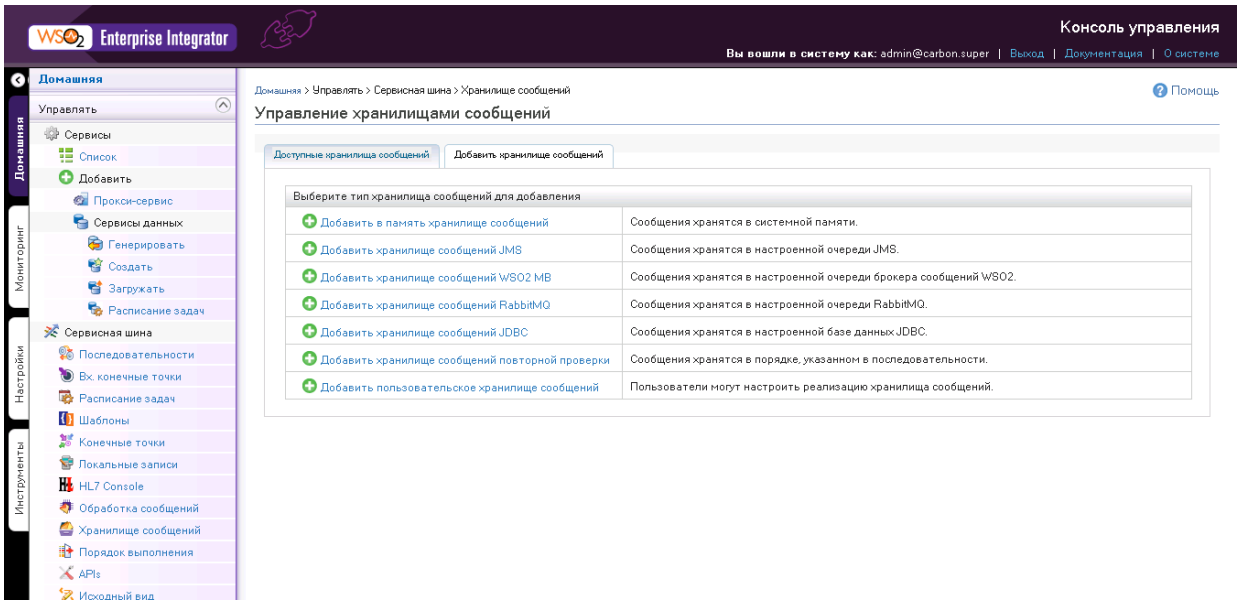


Рисунок 27. Графический интерфейс пункта «Хранилище сообщений».

Хранилища сообщений делятся на семь типов:

- Хранилище сообщений памяти - Сообщения хранятся в системной памяти.
- Хранилище сообщений JMS - Сообщения хранятся в настроенной очереди JMS.
- Хранилище сообщений WS02 MB - Сообщения хранятся в настроенной очереди брокера сообщений WSO2.
- Хранилище сообщений RabbitMQ - Сообщения хранятся в настроенной очереди RabbitMQ.
- Хранилище сообщений JDBC - Сообщения хранятся в настроенной базе данных JDBC.
- Хранилище сообщений повторной проверки - Сообщения хранятся в порядке, указанном в последовательности.
- Пользовательское хранилище сообщений Пользователи могут настроить собственную реализацию хранилища сообщений.

5.1.2.10 « Порядок выполнения »

Порядок выполнения может использоваться с «Enqueue mediator», для выполнения последовательностей с заданным приоритетом. Порядок выполнения используются в сценариях с высокой нагрузкой, когда выполняются разные последовательности для сообщений с разными приоритетами.

Этот подход позволяет контролировать ресурсы, выделенные для выполнения последовательностей, и предотвращать задержку и отбрасывание высокоприоритетных сообщений. Например, сообщение с приоритетом 10 получит в 10 раз больше ресурсов, чем сообщения с приоритетом 1. Установка приоритета гарантирует, что сообщения с высоким приоритетом не будут отброшены.

Для перехода в интерфейс пункта «**Порядок выполнения**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Сервисная шина**» > «**Порядок выполнения**»

Система отобразит графический интерфейс с набором элементов, которые позволяют добавлять и настраивать приоритет (см. Рисунок 28).

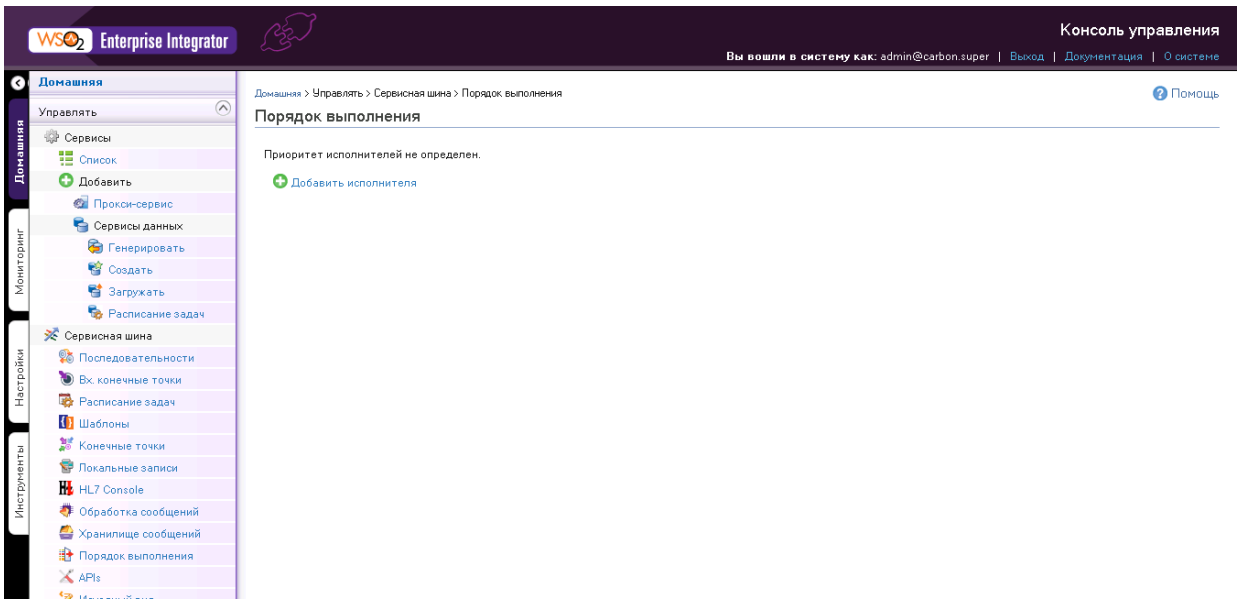


Рисунок 28. Графический интерфейс пункта «Порядок выполнения».

Поле «Дизайн вида приоритетного исполнителя» содержит:

Параметры:

- Имя исполнителя - Имя исполнителя.
- Очереди с фиксированным размером - используются ли очереди с фиксированным размером или нет.
- Max - Максимальное количество потоков в исполнителе.
- Core - базовое количество потоков в исполнителе.
- Keep-Alive - Keep Alive время для потоков.

Очереди:

- Приоритет - приоритет очереди.
- Размер - размер очереди. Этот параметр отображается только в том случае, если выбраны очереди фиксированного размера.

5.1.2.11 «APIs»

API-интерфейсы позволяют отправлять сообщения непосредственно в ESB-профиль, который затем выполняет определенную логику на основе инструкций в вызове HTTP.

Для перехода в интерфейс пункта «**APIs**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Сервисная шина**» > « **APIs** »

Система отобразит графический интерфейс с набором элементов, которые позволяют создавать, изменять и удалять API интерфейсы (см. Рисунок 29).

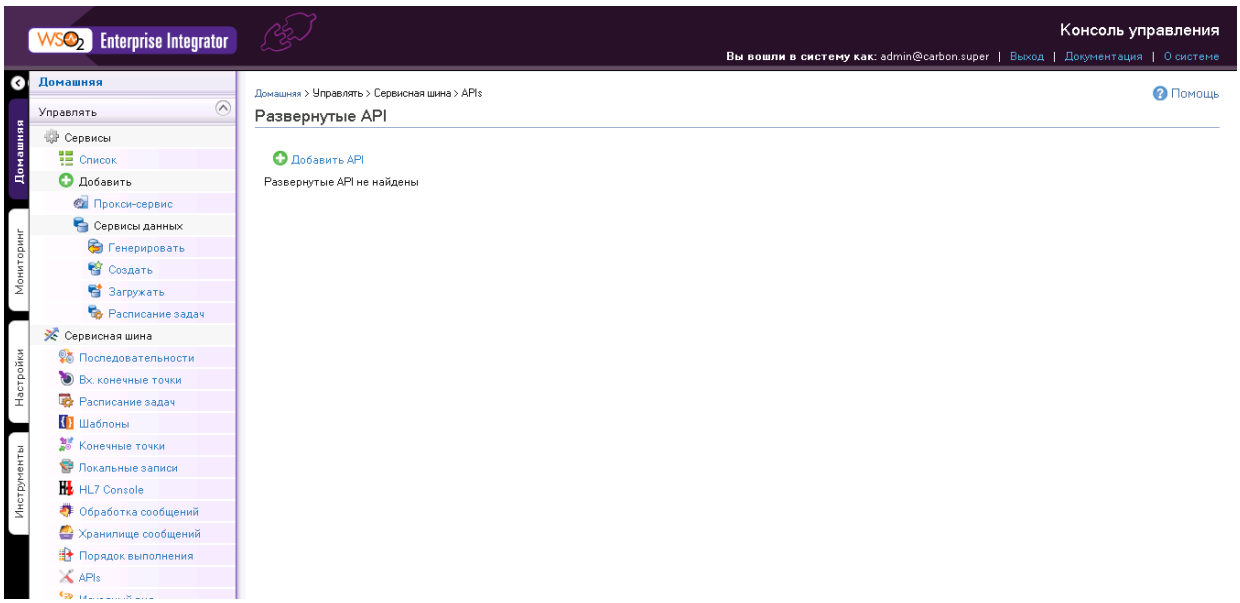


Рисунок 29. Графический интерфейс пункта «APIs».

5.1.2.12 «Исходный вид» (Исходный вид)

Исходный вид предназначена для просмотра конфигурации ESB-профиля в текстовом виде. Данная вкладка не позволяет вносить изменение в конфигурацию.

Для перехода в интерфейс пункта «**Исходный вид**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Сервисная шина**» > «**Исходный вид**»

Система отобразит графический интерфейс с набором элементов, которые позволяют просмотреть исходный вид кода (см. Рисунок 30).

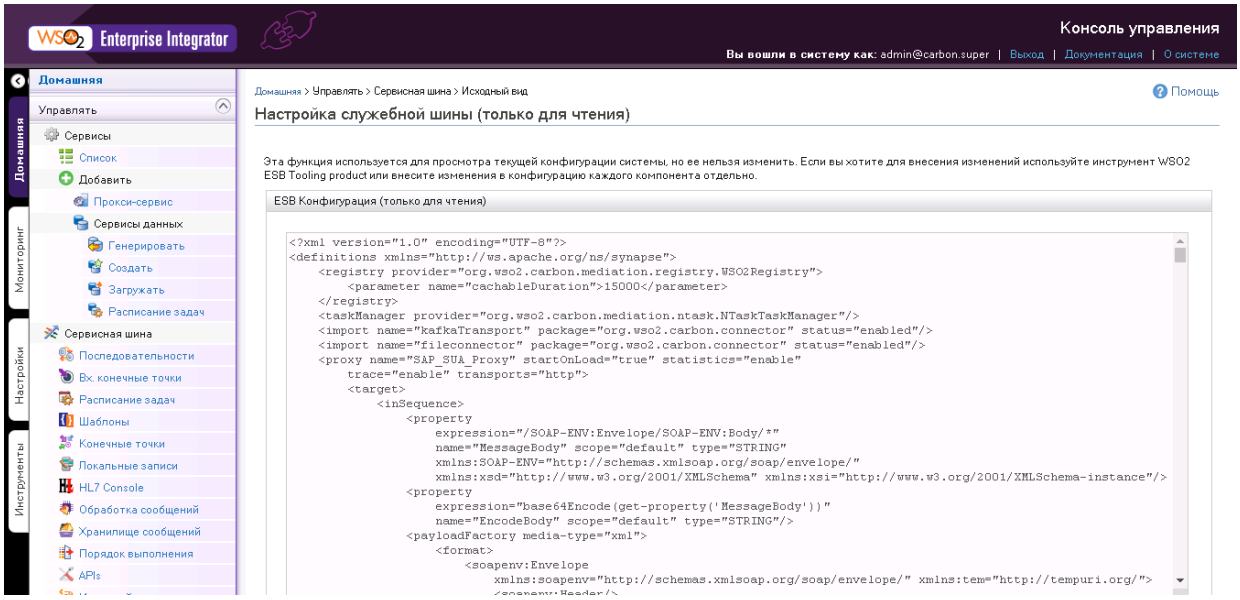


Рисунок 30. Графический интерфейс пункта «Исходный вид».

5.1.3 Раздел «Коннекторы»

Коннектор позволяет взаимодействовать с функциями сторонних продуктов непосредственно из потока сообщений. Предопределенные коннекторы ESB профиля, позволяют подключаться и взаимодействовать с API-интерфейсами различных сервисов. Перечень доступных коннекторов можно получить на портале вендора: https://store.wso2.com/store/assets/esbconnector/Список.

Для перехода в раздел «**Коннекторы**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Коннекторы**»

5.1.3.1 «Список»

Для просмотра установленных коннекторов в систему, выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Коннекторы**» > «**Список**»

Система отобразит графический интерфейс с перечнем установленных коннекторов и набором элементов для управления ими (см. Рисунок 31).

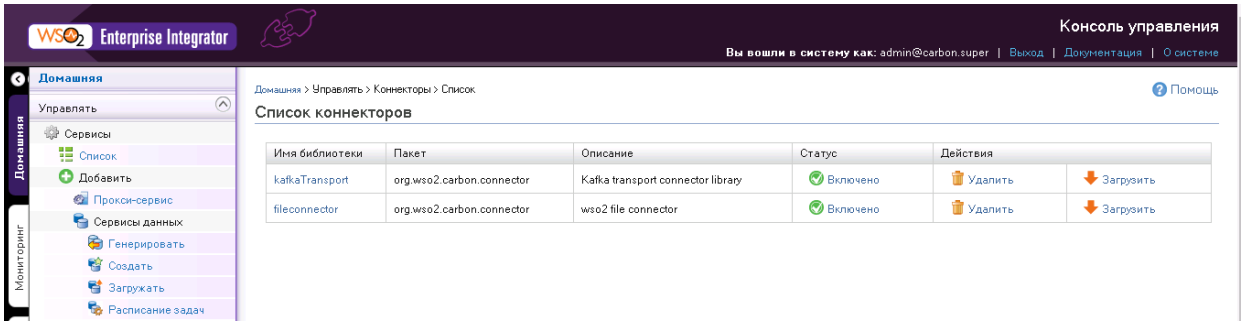


Рисунок 31. Графический интерфейс пункта «Список».

5.1.3.2 «Добавить»

Для добавления коннектора в систему (файл коннектора - zip архив), выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Коннекторы**» > «**Добавить**»

Система отобразит графический интерфейс с набором элементов, которые позволяют создать/добавить коннектор (см. Рисунок 32).

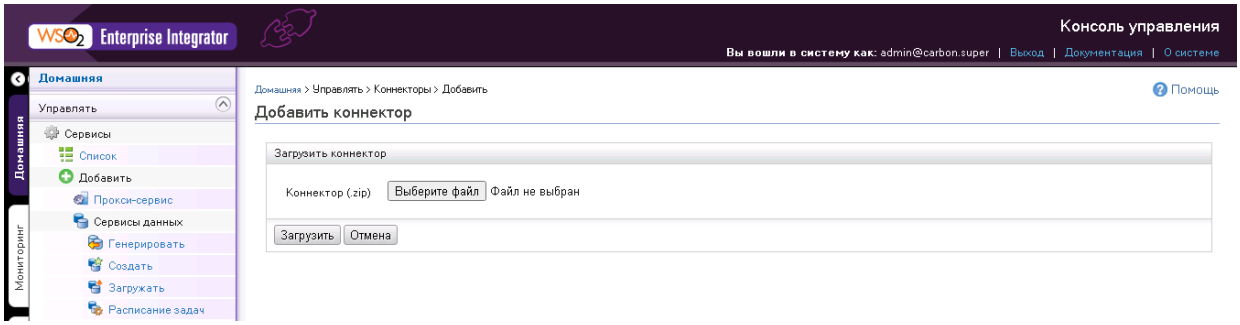


Рисунок 32. Графический интерфейс пункта «Добавить».

5.1.4 Раздел «Средства безопасного Хранения»

«Безопасное хранение» обеспечивает возможность безопасного хранения конфиденциальных данных, таких как текстовые пароли, в файлах платформы WSO2 Carbon: user-mgt.xml, carbon.xml, Axis2.xml, реестр.xml и т. д. Зашифрованные пароли хранятся с сопоставленными псевдонимами. То есть можно использовать псевдонимы вместо реальных паролей в конфигурационных файлах для повышения безопасности.

Например, для некоторых конфигураций требуется логин и пароль администратора. Если пароль пользователя admin - "admin", вы можете использовать псевдоним UserУправлять.AdminUser.Password в вашем конфигурационном файле. Затем сопоставьте этот псевдоним с фактическим паролем "admin". Во время выполнения, продукт будет искать этот псевдоним в безопасном хранилище, а затем расшифрует и использует пароль.

Для перехода в раздел «**Средства безопасного хранения**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Средства безопасного хранения**»

5.1.4.1 «Управление паролями»

Для перехода в интерфейс пункта «**Управление паролями**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Secure Vault Tool**» > «**Управление паролями**»

Система отобразит графический интерфейс с набором элементов, которые позволяют создавать и шифровать новый пароль (см. Рисунок 33).

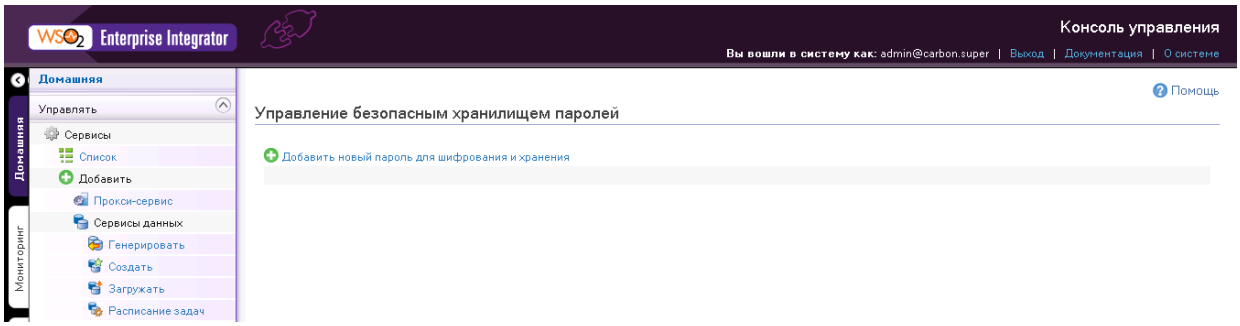


Рисунок 33. Графический интерфейс пункта «Управление паролями».

5.1.5 Раздел «Приложения Carbon»

Carbon приложение - это набор артефактов, которые можно развернуть в консоли управления Carbon, что позволяет перенести решение на основе веб-служб из одной среды в другую.

Артефакт – это объект на уровне платформы, представляет собой замещаемую часть системы. Может быть различных типов: программный скрипт, файл с исходным кодом, исполняемый модуль, источник данных и т. п.

При разработке решений с использованием платформы WSO2 в ней может использоваться огромное количество типов артефактов. Артефакты разрабатываются в среде разработки (WSO2 Developer Studio), а затем перемещаются один за другим в тестовую и промышленную среды.

Развертывание решения, которое содержит множество артефактов, может занять много времени и усилий. Концепция Carbon приложений, решает эту проблему, предоставляя возможность экспортировать все решение как единый артефакт формата (.car), развернув его на сервере приложений через консоль управления.

Для перехода в раздел «Приложения Carbon», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Приложения Carbon**»

5.1.5.1 «Список»

Для перехода в интерфейс пункта «**Список**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Приложения Carbon**> «**Список**»

Система отобразит графический интерфейс с перечнем установленных приложений и набором элементов для управления ими (см. Рисунок 34).

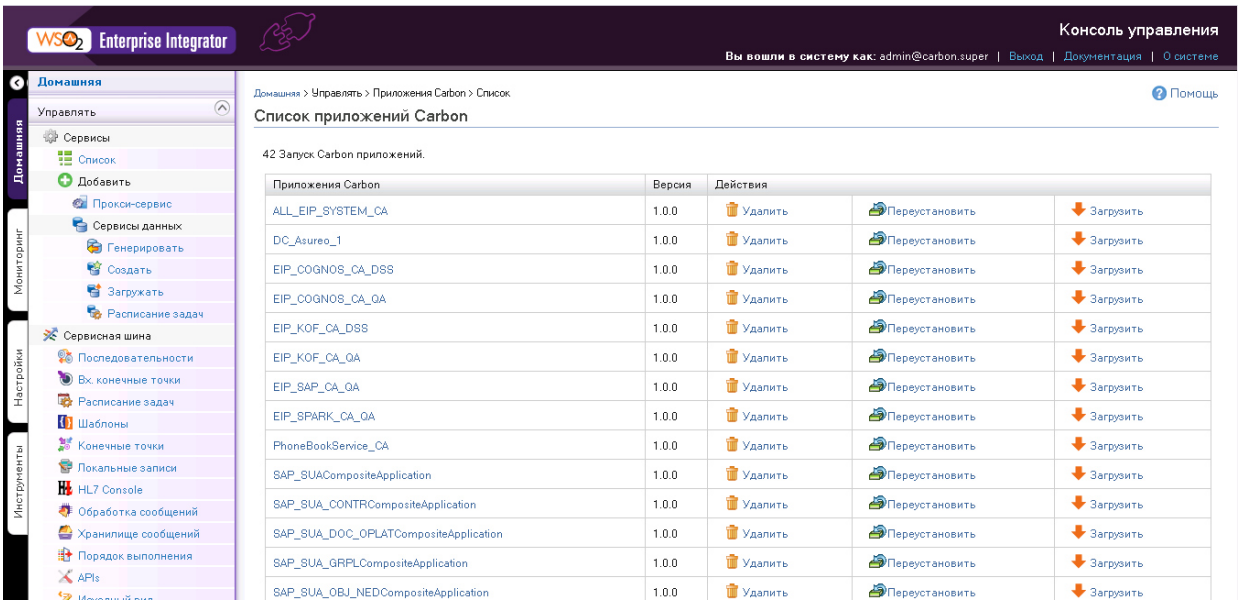


Рисунок 34. Графический интерфейс пункта «Список».

5.1.5.2 «Добавить»

Предварительно экспортируйте файл с расширением (.car) в файловую систему из Developer Studio, для последующего его использования. (см. «Инструкцию по развертыванию ИР»).

Для перехода в интерфейс пункта «**Добавить**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Приложения Carbon**> «**Добавить**»

Система отобразит графический интерфейс с набором элементов, которые позволяют добавить новое приложение. В интерфейсе нажмите кнопку «Выберите файл» (см. Рисунок 35), в открывшемся списке выберите нужный файл с расширением (.car), далее нажмите кнопку «Open» (см. Рисунок 36).

.

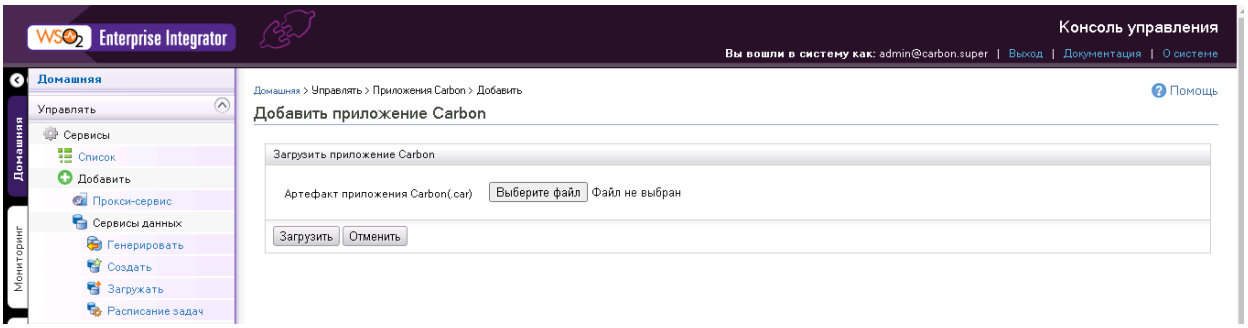


Рисунок 35. Графический интерфейс пункта «Добавить».

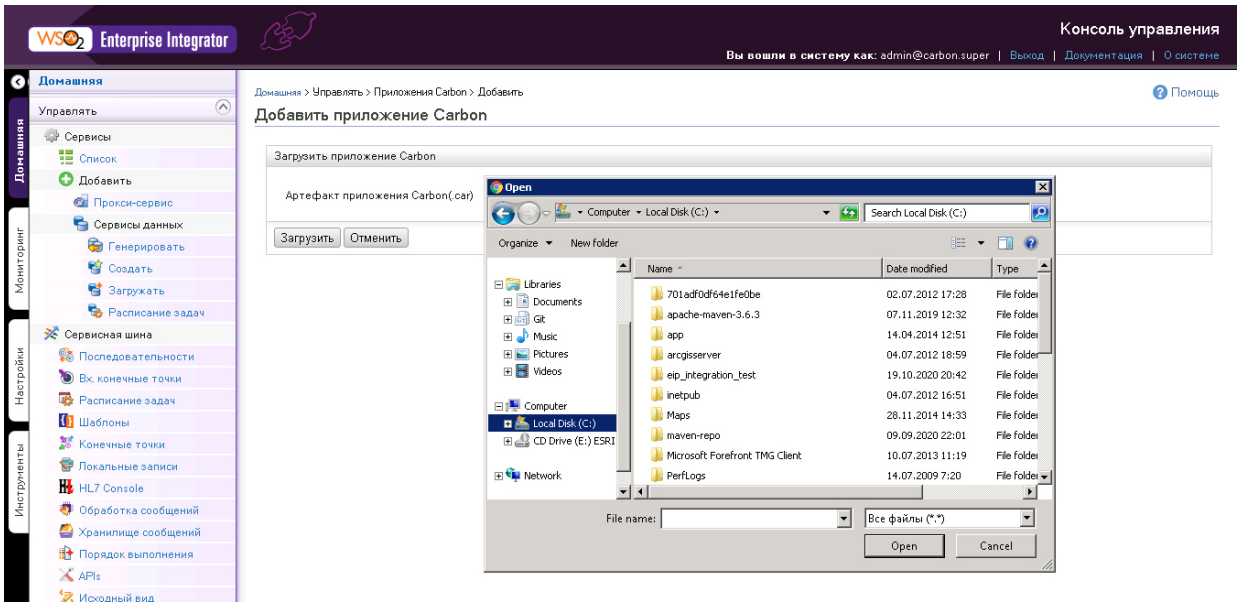


Рисунок 36. Графический интерфейс пункта «Добавить» - выбор файла приложения (.car) для последующей загрузки.

При успешной загрузке, сервис отобразится в поле «Приложения Carbonв графическом интерфейсе пункта «Список».

Каждое ИР может состоять из нескольких Carbon приложений, которые нужно поочередно развернуть на целевой системе в соответствии с инструкцией (см. «Интеграция ЕИП с <Наименование_ИР>», «п.5 Инструкция по особенностям разворачивания ИР»).

5.1.6 Раздел «Модули»

Модуль - это архив с расширением (.mar), который связывает набор классов, библиотек и зависимостей сторонних библиотек. Модули расширяют возможности платформы. Глобальные модули влияют на все службы, развернутые на сервере.

Addressing	4.4.0	Это `WS-Добавитьressing` реализация на Axis2, поддерживающая `WS-Добавитьressing 1.0`
rahas	1.61-wso2v14	Этот модуль используется для включения STS для обслуживания, где он добавляет `RequestSecurityToken` операция с сервисом, к которому подключен модуль.
rampart	1.61-wso2v14	Этот модуль обеспечивает WS-Security и WS-SecureConversation функциональные возможности для Axis2, основанные на Apache WSS4J, Apache XML-Security и Apache Rahas реализации.
relay	4.4.1	Разворачивает двоичные сообщения, поступающие от ретранслятора сообщений для служб администрирования.

WSO2 ESB развернуты следующие модули:

Для перехода в раздел «**Модули**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Модули**»

5.1.6.1 «Список»

Для перехода в интерфейс пункта «**Список**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Модули**» > «**Список**»

Система отобразит графический интерфейс с перечнем установленных и доступных модулей (см. Рисунок 37). Глобально задействованные модули отображаются под списком всех развернутых модулей.

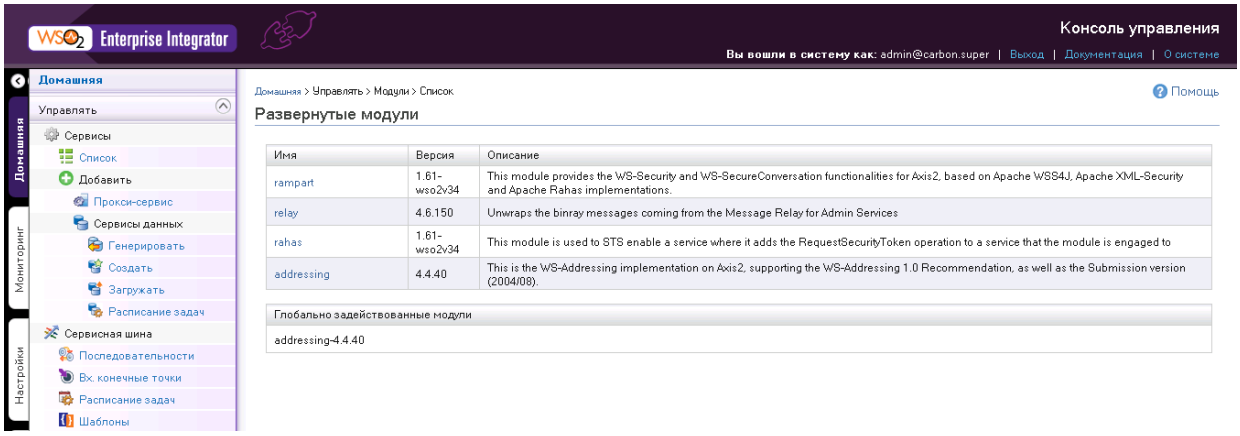


Рисунок 37. Графический интерфейс пункта «Список».

5.1.6.2 «Добавить»

Для перехода в интерфейс пункта «**Добавить**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Модули**» > «**Добавить**»

Система отобразит графический интерфейс с набором элементов, которые позволяют добавить новый модуль. В интерфейсе нажмите кнопку «Выберите файл» (см. Рисунок 38), в открывшемся списке выберите нужный файл с расширением (.mar), далее нажмите кнопку «Open».

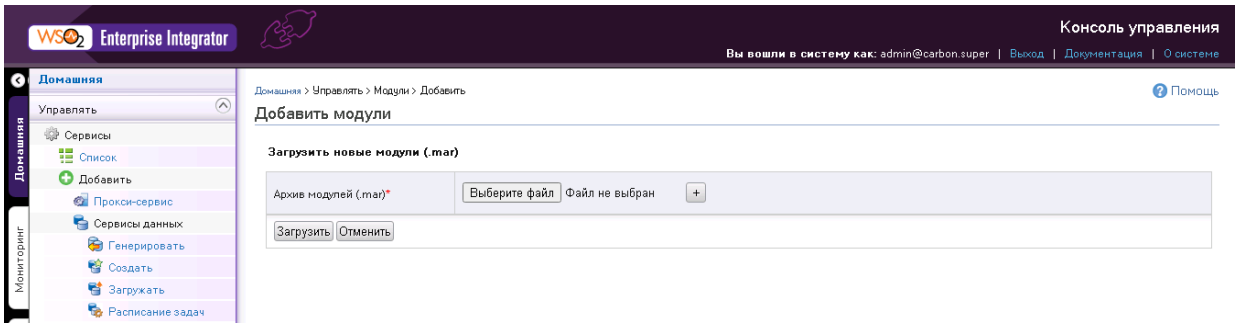


Рисунок 38. Графический интерфейс пункта «Добавить».

5.1.7 Раздел «Разделы»

События - это уведомления, опубликованные в определенном разделе. Веб-службы могут подписываться на разделы, чтобы получать опубликованные в ней события. Это позволяет веб-службам публиковать и подписываться на уведомления друг друга.

Для перехода в раздел «**Разделы**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Разделы**»

5.1.7.1 «Просмотр»

Для перехода в интерфейс пункта «**Browse**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Разделы**» > «**Просмотр**»

Система отобразит графический интерфейс с перечнем существующих разделов и набором элементов, которые позволяют добавить подраздел к выделенному разделу, а также редактировать выделенный раздел (см. Рисунок 39).

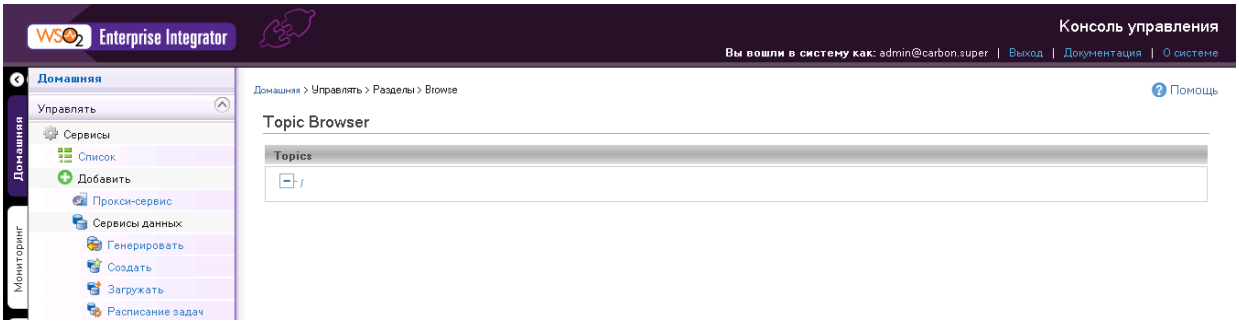


Рисунок 39. Графический интерфейс пункта «Просмотр».

5.1.7.2 «Добавить»

Для перехода в интерфейс пункта «**Добавить**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Разделы**» > «**Добавить**»

Система отобразит графический интерфейс с набором элементов, которые позволяют создать/добавить новый раздел (см. Рисунок 40).

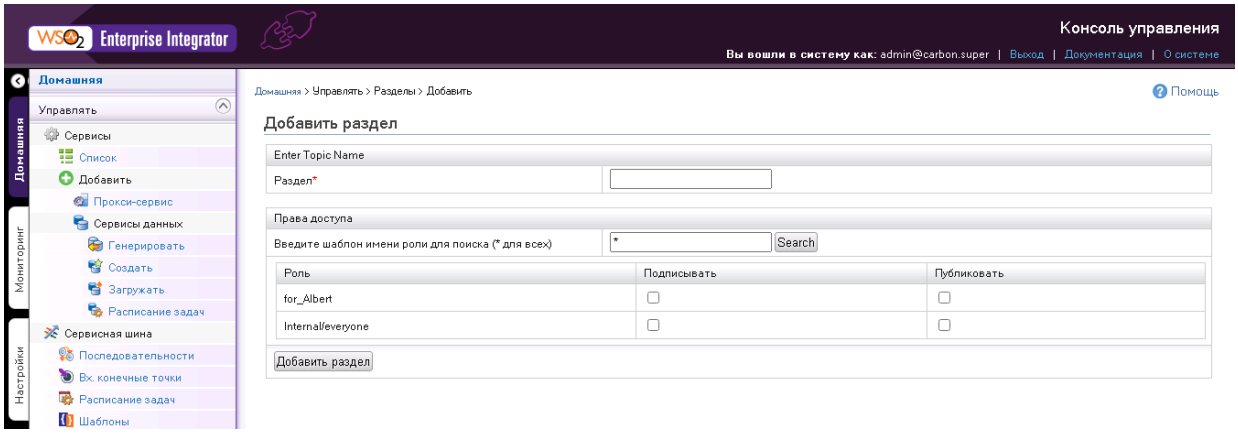


Рисунок 40. Графический интерфейс пункта «Добавить».

5.1.8 Раздел «Выключение/Перезапуск»

Для перехода в раздел «**Выключение/Перезапуск**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Выключение/Перезапуск**»

Система отобразит графический интерфейс с набором элементов, которые позволяют управлять выключением/перезапуском сервера (см. Рисунок 41).

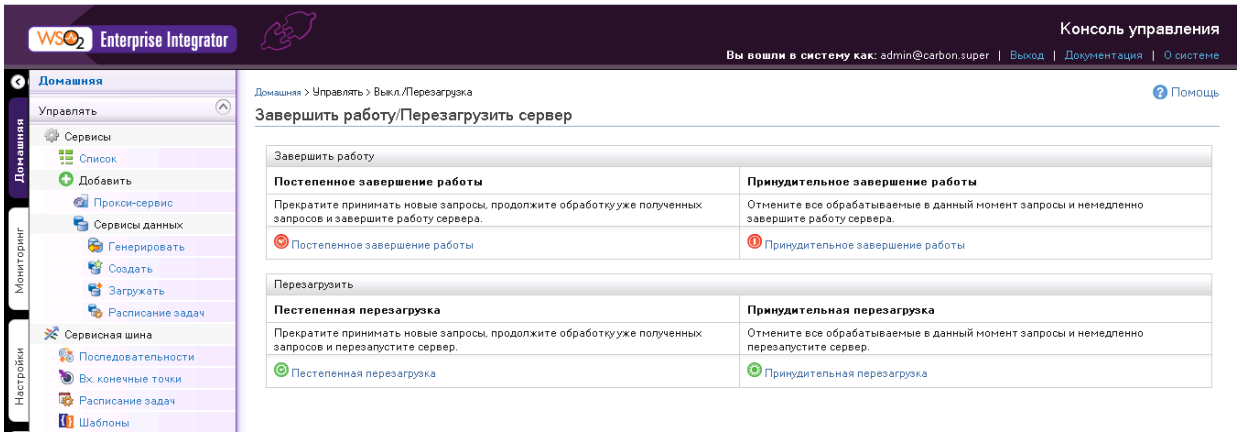


Рисунок 41. Графический интерфейс пункта «Выключение/перезапуск».

Постепенное завершение – прекращение приема новых запросов, завершить обработку уже полученных запросов, а затем выключить сервер.

Постепенная перезагрузка – прекращение приема новых запросов, завершить обработку уже полученных запросов, а затем перезапустить сервер.

Принудительное завершение – прекращение обработки всех текущих запросов и незамедлительно отключить сервер.

Принудительная перезагрузка – прекращение обработки всех текущих запросов и незамедлительно перезапустить сервер.

5.1.9 Раздел «Реестр»

Реестр - это хранилище содержимого и хранилище метаданных для различных артефактов, таких как службы, WSDL и файлы конфигурации. Эти артефакты обозначаются уникальными путями, где путь похож на путь к файлу Unix. В продуктах WSO2 все конфигурации, относящиеся к модулям, ведению журналов, безопасности, источникам данных и другим группам услуг, по умолчанию хранятся в реестре шины.

Реестр хранит ресурсы в структурированном виде.

Область реестра любого продукта WSO2 содержит три основных раздела:

- Локальный репозиторий используется для хранения данных конфигурации и времени выполнения, локальных для сервера. Этот раздел не должен использоваться несколькими серверами. Точка монтирования - /_system/local;
- Репозиторий конфигурации используется для хранения конфигураций, специфичных для продукта. Этот раздел может совместно использоваться несколькими экземплярами одного и того же продукта (например, совместно использовать конфигурации ESB в кластере ESB). Точка монтирования - /_system/config;
- Репозиторий управления используется для хранения конфигурации и данных, которые совместно используются всей платформой. Обычно это службы, описания служб, конечные точки или источники данных. Точка монтирования этого реестра - /_system/governance.

Для перехода в раздел **«Реестр»** выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Реестр**»

5.1.9.1 «Просмотр»

Для перехода в интерфейс пункта «**Просмотр**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Реестр**» > «**Просмотр**»

Система отобразит графический интерфейс с древовидной структурой реестра и набором элементов, которые позволяют осуществлять поиск, просмотр и управление содержимым реестра (см. Рисунок 42; см. Рисунок 43; см. Рисунок 44).

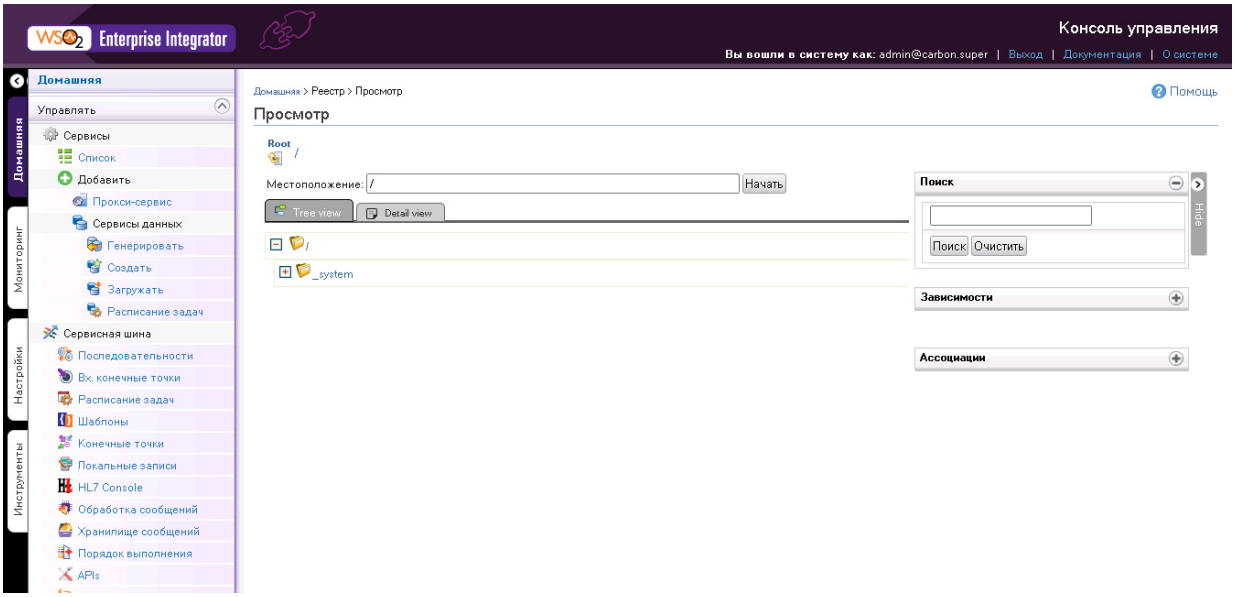


Рисунок 42. Графический интерфейс пункта «Просмотр».

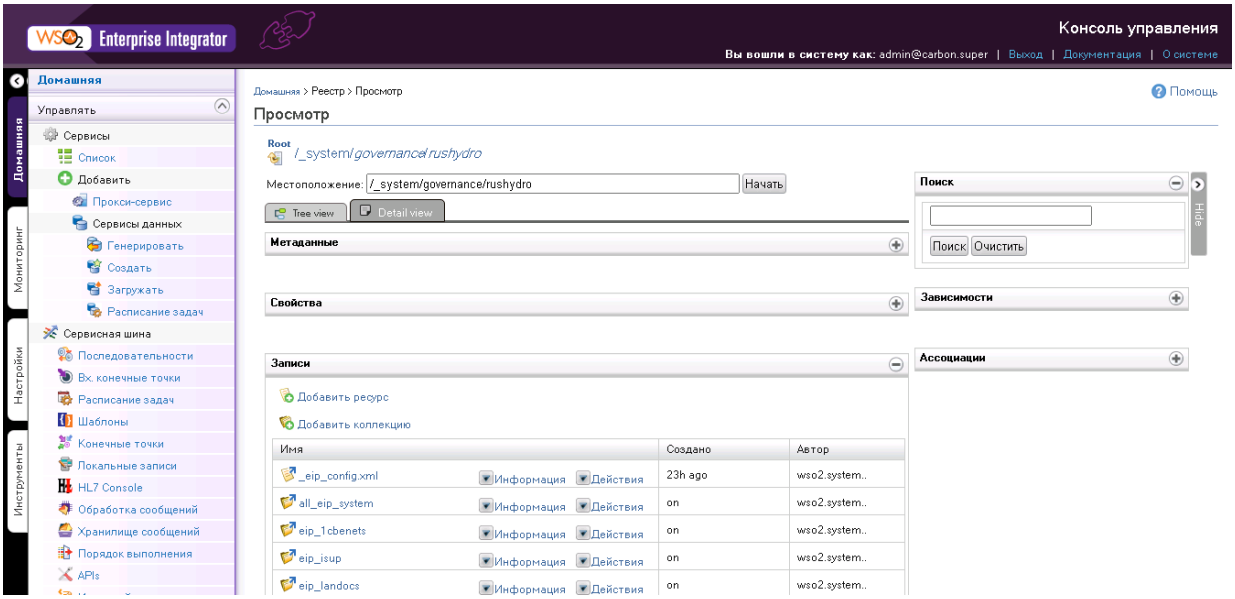


Рисунок 43. Графический интерфейс пункта «Просмотр».

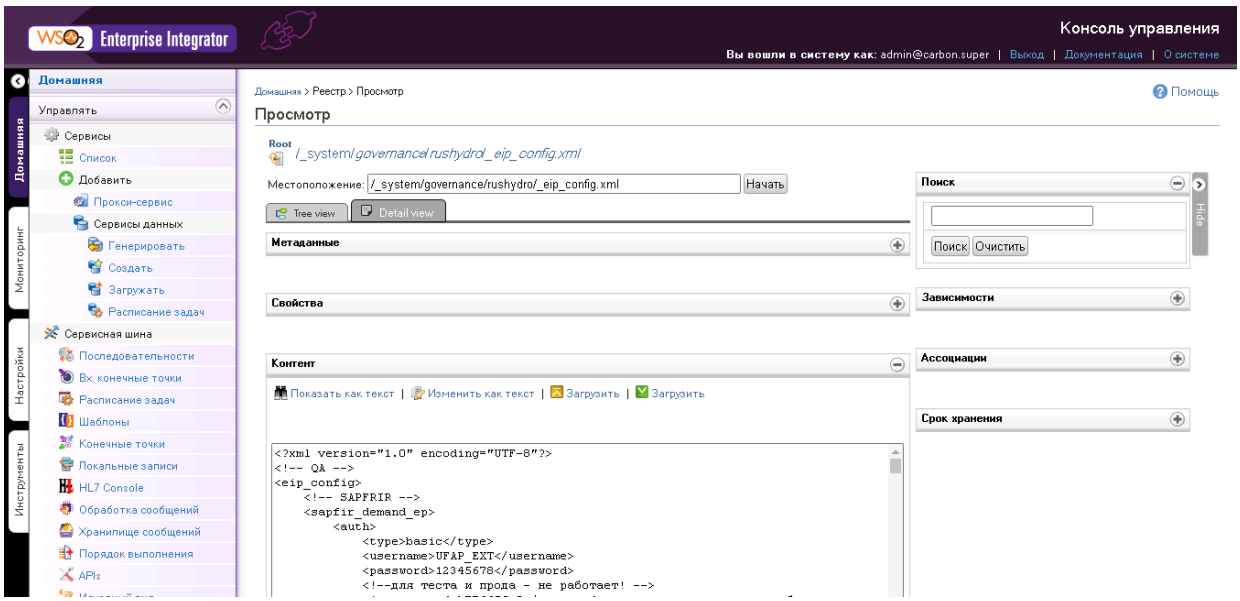


Рисунок 44. Графический интерфейс пункта «Просмотр».

5.1.9.2 «Поиск»

Для перехода в интерфейс пункта «**Поиск**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Реестр**» > «**Поиск**»

Система отобразит графический интерфейс с перечнем существующих разделов и набором элементов, которые позволяют добавить подраздел к выделенному разделу, а также редактировать выделенный раздел (см. Рисунок 45).

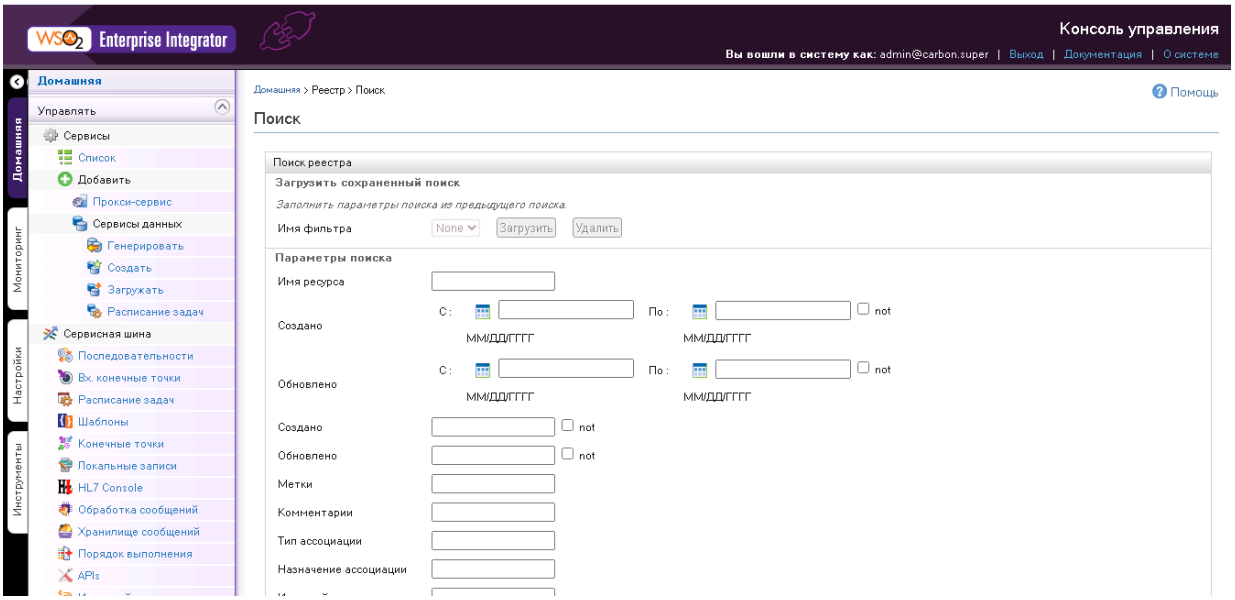


Рисунок 45. Графический интерфейс пункта «Поиск».

5.1.10 Вкладка: «Мониторинг»

WSO2 журналирует различные виды логов. Логирование осуществляется в виде последовательно записываемой информации в текстовые файлы. Основные файлы журналов и файлы для их конфигурации перечислены ниже:

- Carbon log (wso2carbon.log) - это файл журнала, который охватывает все функции управления продуктом. Настройки осуществляются в файле log4j.properties (хранится в каталоге <EI_HOME>/conf).
- Журналы аудита - используются для отслеживания последовательности действий, которые влияют на конкретную задачу, выполняемую на сервере. Также настраивается в файле log4j.properties.
- Журналы HTTP-доступа: HTTP-запросы / ответы регистрируются в журналах доступа для мониторинга действий, связанных с использованием приложения. Настраиваются в файле catalina-server.xml. (хранится в каталоге EI_HOME>/conf/tomcat/).
- Журналы сервисов / событий: это журналы для отслеживания сервисов и событий с использованием отдельного файла журнала (wso2- <product> - trace.log). Если предусмотрен сервисом - настраивается в соответствии с инструкцией.

Настройки журналирования, такие как периодичность, избыточность и т.д. настраиваются в соответствующих основных файлах, указанных выше.

Захват системных данных при возникновении инцидентов

Carbon Dump - инструмент, включенный в поставку WSO2, для сбора всех необходимых данных (дампы кучи и потоков) во время возникновения инцидента. Carbon Dump создает ZIP-архив с собранными данными, который поможет проанализировать данные системы и определить проблему, вызвавшую ошибку. Поэтому рекомендуется запускать этот инструмент при возникновении инцидентов. Если кластер продуктов WSO2 выходит из строя из-за истощения ресурсов, дампы кучи и потока укажут на причину. Зачастую исчерпание ресурса может произойти по двум причинам:

- Из-за ошибки в системе.
- Фактическое ограничение ресурсов на основе низких значений конфигурации.

Файл Carbon Dump располагается в системе: <EI_HOME>/carbondump.sh. При использовании необходимо указать идентификатор процесса (pid) экземпляра Carbon и <EI_HOME>.

Пример запуска команды:

```
sh carbondump.sh -carbonHome /opt/wso2ei-6.4.0/ -pid 5151
```

Архив создается в директории, в которой была вызвана команда. Архив можно скопировать с сервера для подробного изучения на локальной машине.

Примечание: Куча – динамически выделяемая область памяти в операционной системе под нужды приложения. Java Heap (куча) используется Java Runtime для выделения памяти под объекты и JRE классы.

5.1.11 «Журналы приложений»

Журналы приложений - это события, которые записываются при их вызове приложением или программой, запущенной в системе. Журналы приложений запущенного экземпляра Carbon отображают события его развернутых веб-приложений и веб-служб. Страница «**Журналы приложений»** представляет детальное описание журналов приложений, функционал интерфейса позволяет пользователю Выберите конкретное приложение и просматривать журналы относящиеся к выбранному приложению.

Для перехода в интерфейс пункта «**Журналы приложений**», выполните следующие действия:

Выберите вкладку «**Мониторинг**» > «**Журналы приложений**»

Система отобразит графический интерфейс с набором элементов, которые позволяют осуществлять поиск, просмотр содержимого системных журналов и делать выборку по уровню критичности сообщений (см. Рисунок 46).

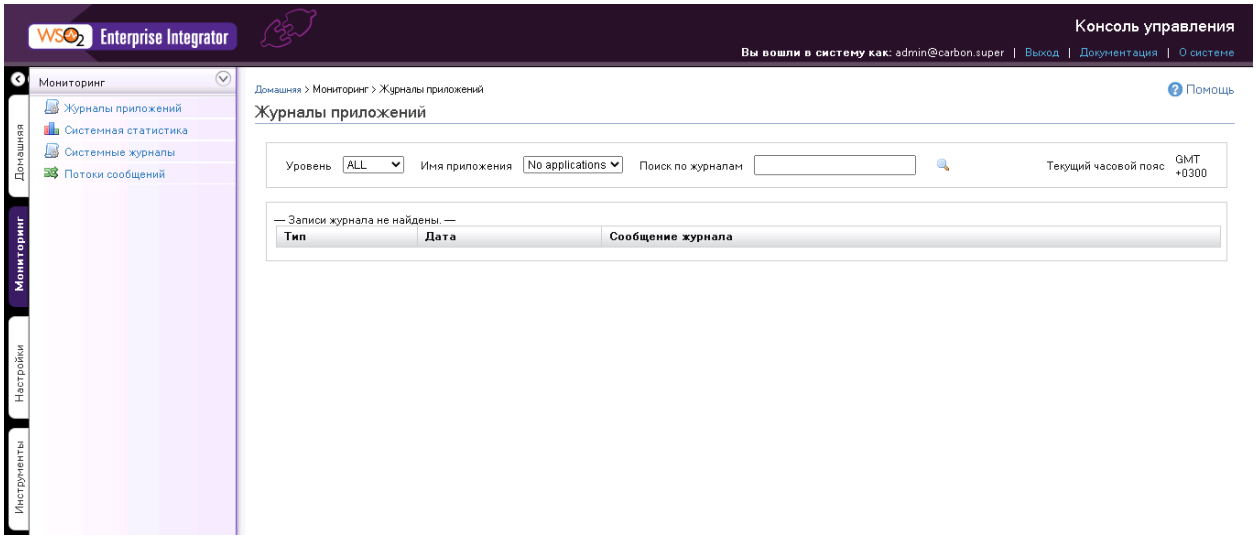


Рисунок 46. Графический интерфейс пункта «Журналы приложений».

Категории уровня критичности системных сообщений (логи):

TRACE - трассировка сообщений;

DEBUG - отладочные сообщения;

INFO - информационные сообщения;

WARN - предупреждающие сообщения;

ERROR - сообщения об ошибках;

FATAL - сообщения о фатальных ошибках;

ALL - отображает все категории журналов.

На странице пункта «**Журналы приложений**» отображается информация о файлах журналов текущего продукта. Файлы журналов можно получить двумя способами:

- Если syslog-ng настроен, файлы журналов берутся из удаленного расположения, где они размещаются с помощью сервера syslog-ng.
- Если syslog-ng не настроен, файлы журналов берутся из локальной файловой системы (супертенант или автономные приложения).

5.1.12 «Системная статистика »

Консоль управления позволяет отслеживать статистику производительности на работающем экземпляре Carbon. Эти статистические данные включают информацию о наличии памяти, количестве запросов, имени сервера, времени запуска сервера, времени работы системы, активных службах, общем объеме памяти, среднем, минимальном, максимальном времени ответа и т. д.

Для перехода в интерфейс пункта «**Системная статистика**», выполните следующие действия:

Выберите вкладку «**Мониторинг**» > «**Системная статистика**»

Система отобразит графический интерфейс, отображающий статистическую информацию и набором элементов, который позволяет настраивать параметры статистики (см. Рисунок 47; Рисунок 48).

Статистические данные, включают информацию по следующим позициям:

- Информация о сервере
- Информация о количестве запросов;
- Информация о среднем, минимальном, максимальном времени отклика;
- График памяти;
- Количество активных сервисов.

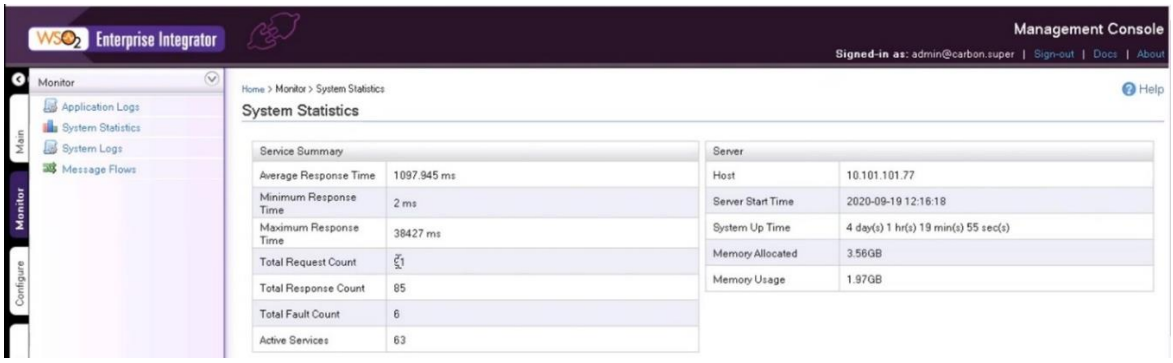


Рисунок 47. Графический интерфейс пункта «Системная статистика».

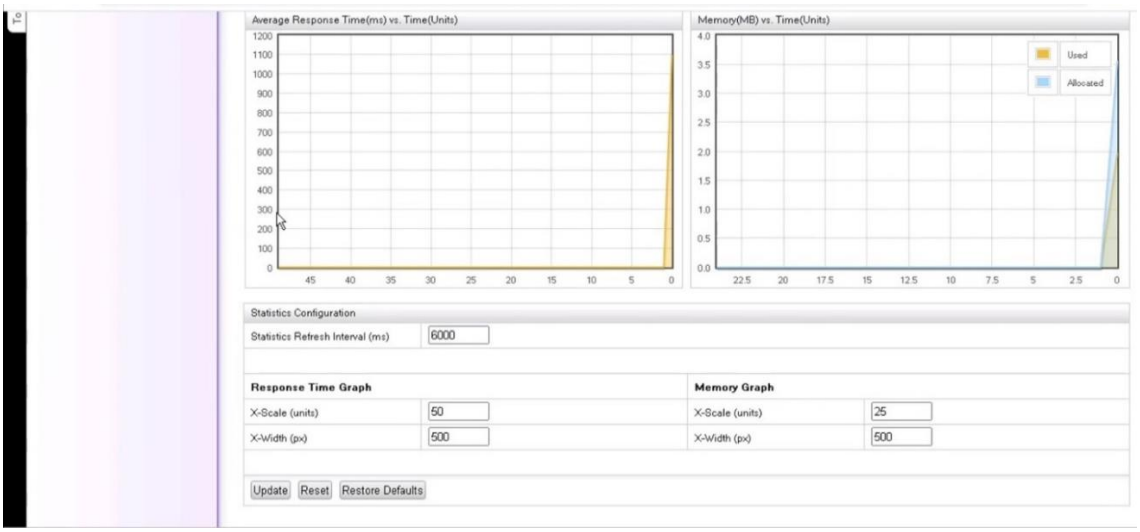


Рисунок 48. Графический интерфейс пункта «Системная статистика».

Для просмотра статистики через панель управления профиля аналитики, необходимо перейти по адресу https://ei-domain-name:9643/portal (см. Рисунок 49).

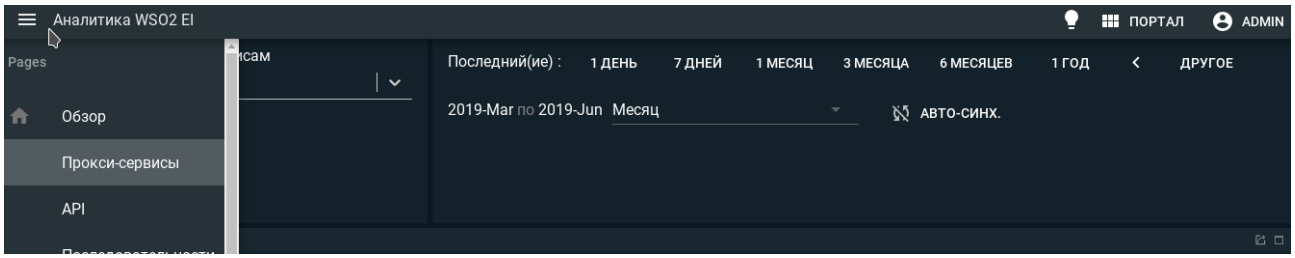


Рисунок 49. Графический интерфейс «Прокси-серверы».

Для продуктивного контура:

```
https://ei-domain-name:9643/portal
```

1. Авторизуйтесь, используя учетные данные.
2. Чтобы просмотреть статистику для определенного диапазона дат, выберите нужный диапазон дат в правой верхней строке меню.
3. Чтобы просмотреть статистику прокси-сервиса, выберете соответствующий прокси-сервис из списка.
4. Панель управления аналитики отобразит статистику, относящуюся к выбранному прокси-сервису.

Отображаемая информация предоставит данные за выбранный период дат:

- общий процент успешных и неудачных попыток;
- количество успешных / неудачных запросов;
- задержку сообщений;
- список сообщений, обработанных в течение выбранного промежутка времени;
- поток сообщений, отправленный в прокси-сервис.

5.2 Мониторинг SOAP сообщений. Включение статистики / отслеживания сообщений.

Встроенный инструмент Tracer, позволяет отображать в журналах все SOAP сообщения, включая запросы сообщений и ответы, которые проходят через систему. Трассировка включается на отдельно выбранный сервис. По умолчанию данная опция выключена и не рекомендуется включать трассировку в производственных средах, поскольку она генерирует большое количество событий, которые снижают производительность системы. Поэтому рекомендуется использовать трассировку только в среде разработки.

Файл synapse.properties, располагается в <EI_HOME>/conf/. Ниже указаны параметры по умолчанию:

```
# Configuration to enable mediation flow analytics
mediation.flow.statistics.enable=false
mediation.flow.statistics.tracer.collect.payloads=false
mediation.flow.statistics.tracer.collect.properties=false
mediation.flow.statistics.event.consume.interval=1000
mediation.flow.statistics.event.clean.interval=15000

# Configuration to enable statistics globally irrespective of the individual artifact level setting
mediation.flow.statistics.collect.all=false

# Script Mediator Pool (impatcts only external scripts)
Приложение 1    #synapse.script.mediator.pool.size=15
```

mediation.flow.statistics.enable	значение true включит статистику: Время, проведенное на каждом посреднике. Время, затраченное на обработку каждого сообщения. Количество ошибок одного потока сообщений.
mediation.flow.statistics.tracer.collect.payloads	значение true включит трассировку payloads
mediation.flow.statistics.tracer.collect.properties	значение true включит трассировку: Свойства контекста сообщения. Свойства области сообщения.
mediation.flow.statistics.collect.all	включит полную статистику для всех артефактов

Включение статистики synapse.properties позволяет отслеживать статистику всех артефактов ESB (прокси-сервисы, REST API, последовательности и конечные точки) по умолчанию. Кроме того, у вас есть возможность включить статистику только для требуемых артефактов. Для этой цели вы можете использовать консоль управления.

В кластерном развертывании статистика должна быть включена для требуемых артефактов во всех узлах EI.

Для включения статистики по требуемому прокси-сервису, выполните следующие действия

- Выберите вкладку «**Домашняя**» > «**Управлять**» > «**Сервисы**» > «**Список**»
- Выберите требуемый прокси-сервис (см. Рисунок 50);

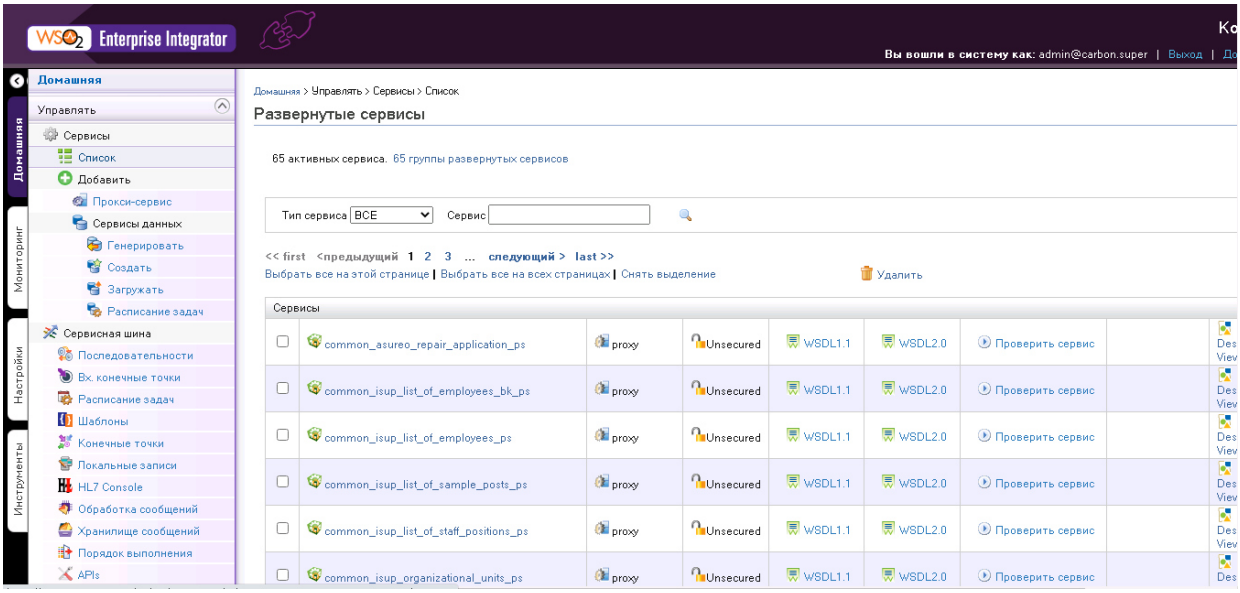


Рисунок 50. Графический интерфейс пункта «Список».

- Перейдите в сервисную панель (см. Рисунок 51) и включите статистику и трассировку и нажмите «ОК» в появившемся уведомлении, чтобы подтвердить, что статистика и трассировка включена. Функция трассировки - отслеживает сообщение и фиксирует его состояние, т.е. происходит отслеживание с целью мониторинга того, как сообщение обрабатывается и какие этапы проходит.

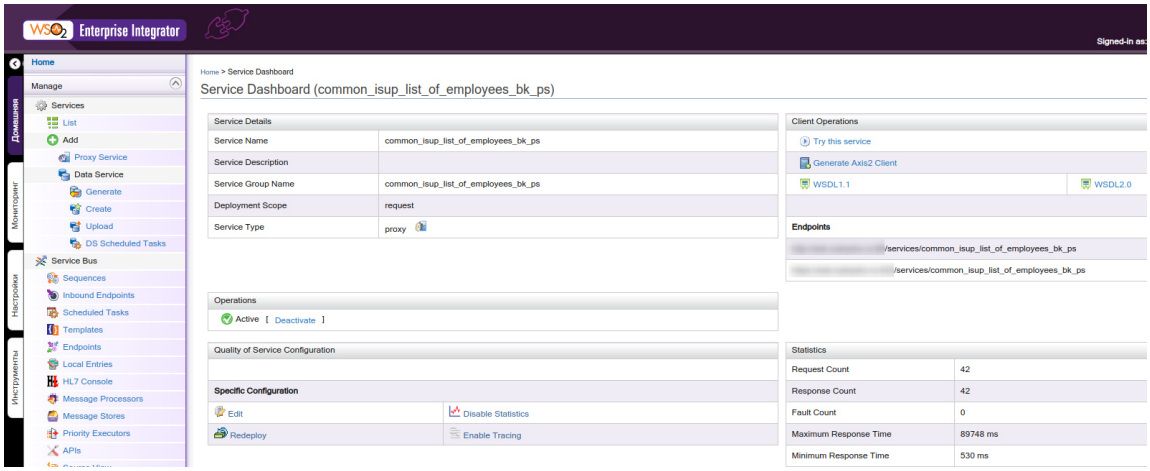


Рисунок 51. Графический интерфейс пункта «Сервисная панель»

Ниже указан перечень артефактов, по которым можно включить статистику и трассировку, а также провести анализ:

- Прокси сервис;
- REST API;
- Последовательность;
- Конечные точки.

5.2.1 «Системные журналы»

Системные журналы отображают информацию о всех события запущенного экземпляра Carbon. Файлы журналов можно получить двумя способами:

Если System Logs настроен, файлы журналов берутся из внешней среды, где файлы журналов размещаются с помощью сервера syslog-ng. Расположение файлов журнала на диске указывается в log4j.properties файле.

Если syslog-ng не настроен, файлы журналов берутся из локальной файловой системы (супертенант или автономные приложения).

Для перехода в интерфейс пункта «**Системные журналы**», выполните следующие действия:

Выберите вкладку «**Мониторинг**» > «**Системные журналы**»

Система отобразит графический интерфейс, отображающий записи системного журнала и набором элементов, который позволяет осуществить поиск записей и делать выборку по уровню критичности сообщений (см. Рисунок 52).

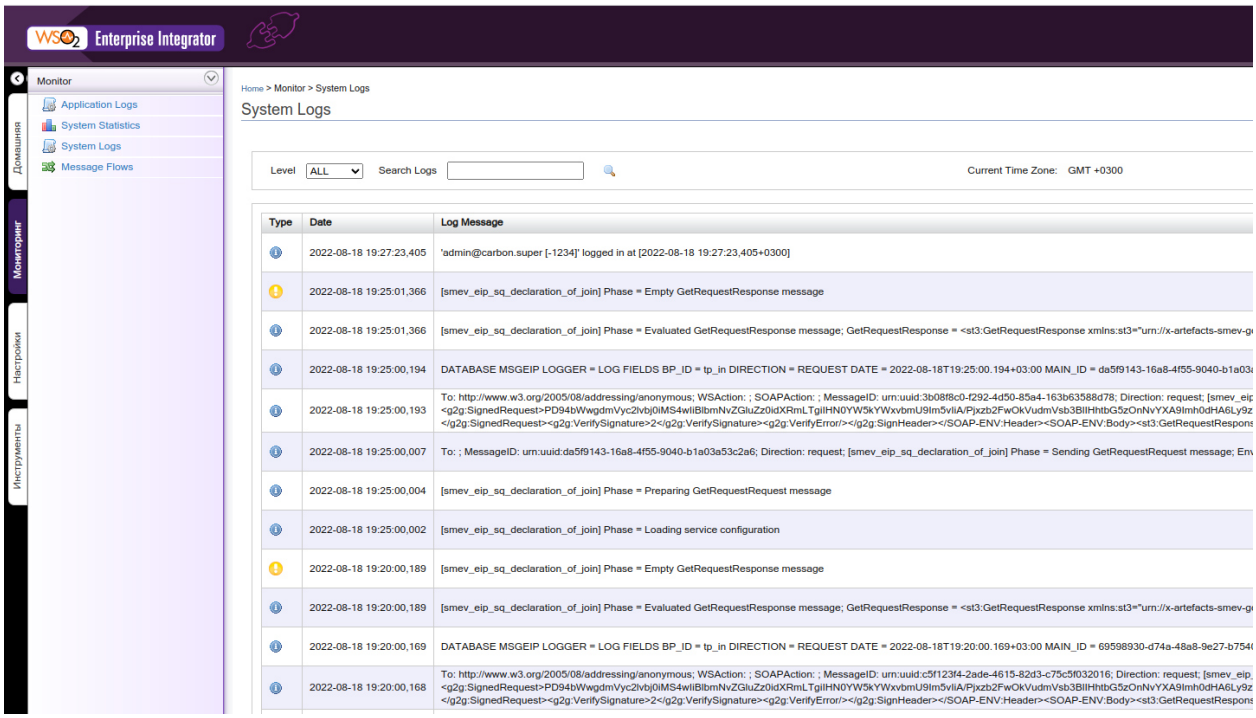


Рисунок 52. Графический интерфейс пункта «Системные журналы».

Доступные категории записей для просмотра и выбора:

- **TRACE** - трассировка сообщений;

- **DEBUG** - отладочные сообщения;
- **INFO** - информационные сообщения;
- **WARN** - предупреждающие сообщения;
- **ERROR** - сообщения об ошибках;
- **FATAL** - сообщения о фатальных ошибках.

Также можно произвести настройки отображения уровня избыточности логов Carbon и изменить существующую настройку конфигурации, для этого воспользуйтесь элементами интерфейса пункта «**Регистрация**»: «**Домашняя**» > «**Настройки**» > «**Регистрация**» (см. Рисунок 53).

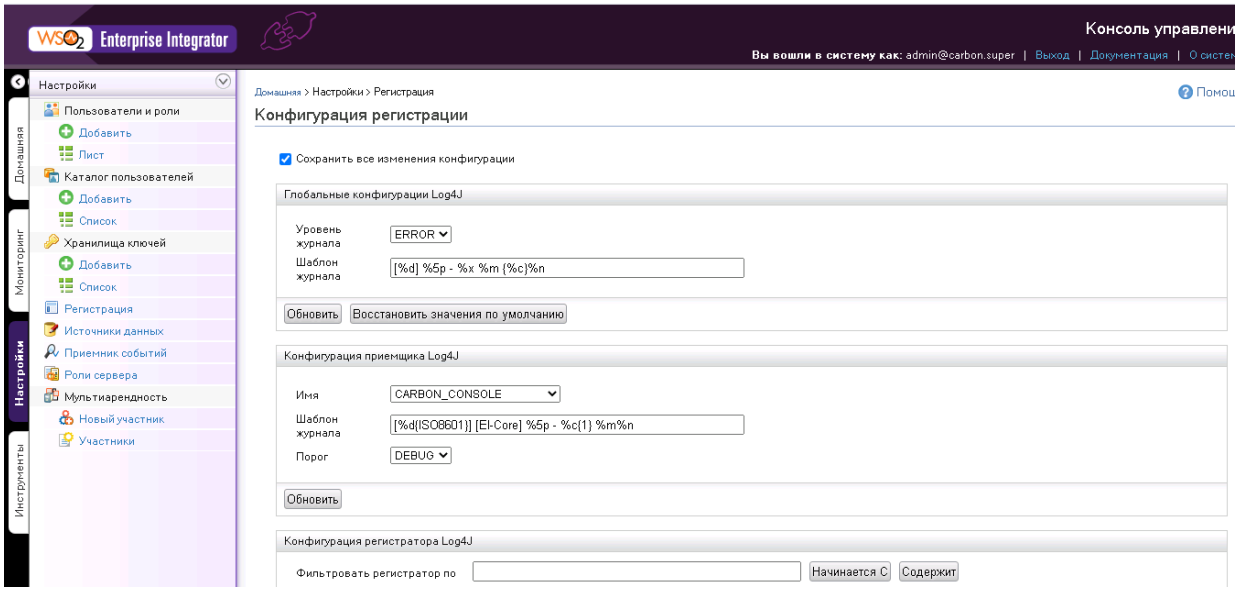


Рисунок 53. Графический интерфейс пункта «Регистрация».

5.2.2 «Потоки сообщений»

Раздел Просмотр Потоков на вкладке мониторинг отображает потоки сообщений, которые обеспечивают графическое или текстовое представление глобально задействованных обработчиков в системе в данный момент времени.

Для перехода в интерфейс пункта «**Потоки сообщений**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Мониторинг**» > «**Потоки сообщений**»

Система отобразит графический интерфейс с схемами потоков сообщений (см. Рисунок 54).

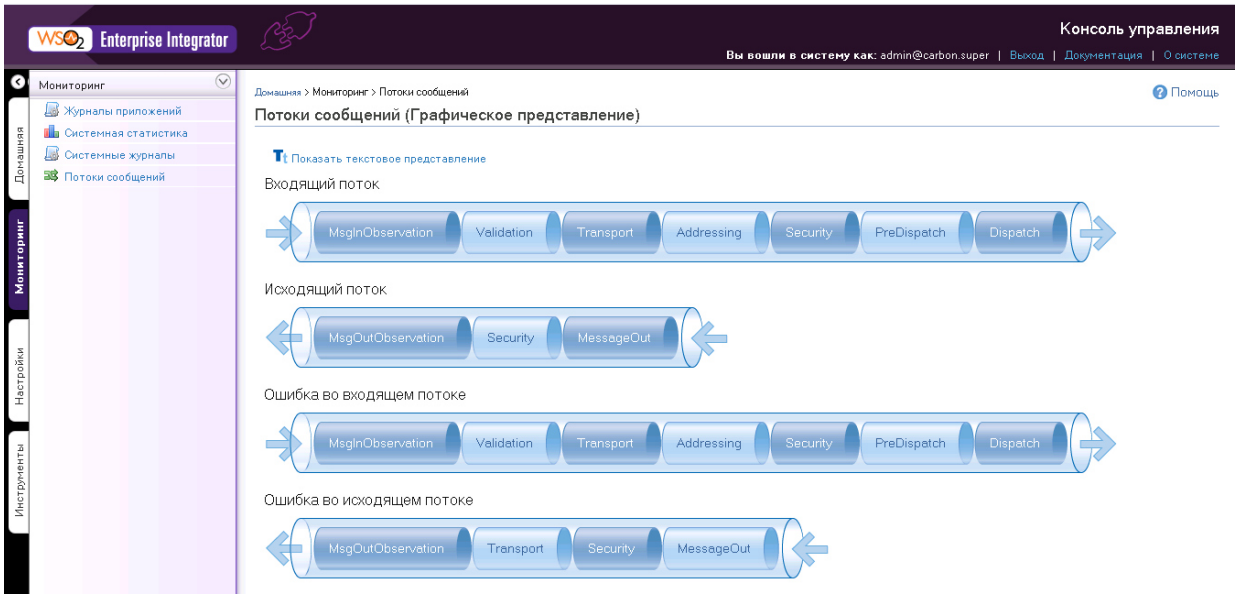


Рисунок 54. Графический интерфейс пункта «Потоки сообщений».

5.3 Вкладка: «Настройки»

5.3.1 Раздел «Пользователи и Роли»

WSO2 поддерживает модель аутентификации на основе ролей, в которой права пользователя зависят от прикрепленной роли. Каждая роль может быть настроена с минимальным или максимальным количеством доступов и прав. Набор доступов и прав, принадлежащих пользователю, определяется ролями, назначенными этому пользователю. Если пользователю назначено несколько ролей, их права объединяются.

По умолчанию для всех продуктов WSO2 настроены следующие роли:

- Администратор - обеспечивает полный доступ ко всем функциям и элементам управления. По умолчанию, пользователь с правами администратор обладает максимальным уровнем доступа и правами.
- Внутренний / Каждый - каждому новому пользователю назначается эта роль по умолчанию.
- Внутренний / Система - эта роль не отображается в консоли управления.

Можно настроить роли, в зависимости от типа функций, установленных в вашем продукте. Например, в сервере хранения WSO2 (на котором установлены функции Cassandra и RSS Управлятьr) по умолчанию также будут определены следующие роли: Internal / Cassandra и Internal / RSSУправлятьr.

Для перехода в раздел «**Пользователи и Роли**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Настройки**» > «**Пользователи и Роли**»

5.3.1.1 «Добавить»

Для перехода в интерфейс пункта «**Добавить**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Настройки**» > «**Пользователи и Роли**» > «**Добавить**»

Система отобразит графический интерфейс с набором элементов, которые позволяют добавить новых пользователей или добавить новую роль, а также импортировать данные с информацией о пользователе (см. Рисунок 55; см. Рисунок 56)

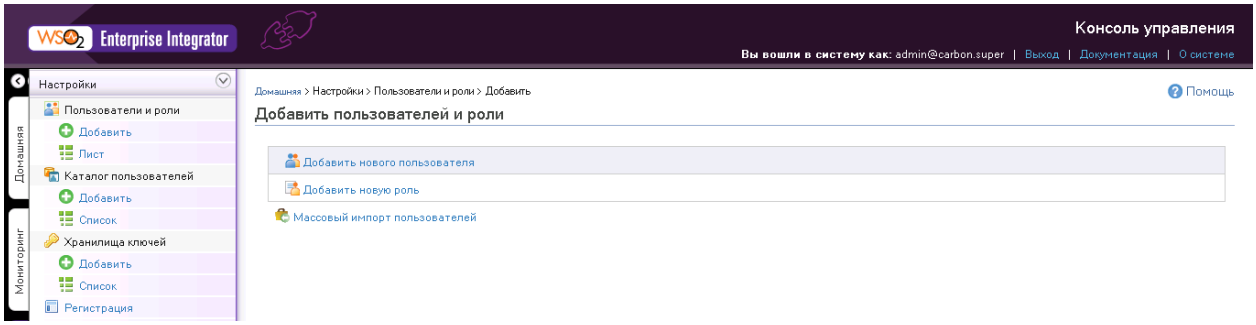


Рисунок 55. Графический интерфейс пункта «Добавить».

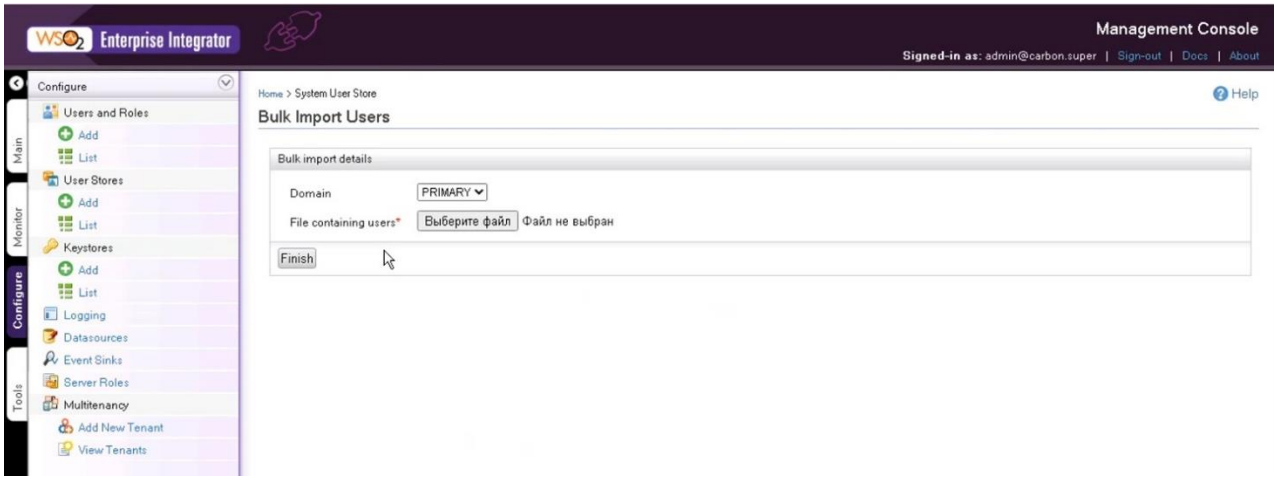


Рисунок 56. Графический интерфейс пункта «Добавить», вкладка «System User Store».

3.3.1.2 «Список» (Список)

Для перехода в интерфейс пункта «**Список**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Настройки**» > «**Пользователи и Роли**» > «**Список**»

Система отобразит графический интерфейс с набором элементов, которые позволяют просмотреть зарегистрированных пользователей и назначенные им роли, сменить пароль, роль или удалить профиль пользователя (см. Рисунок 57; см. Рисунок 58).

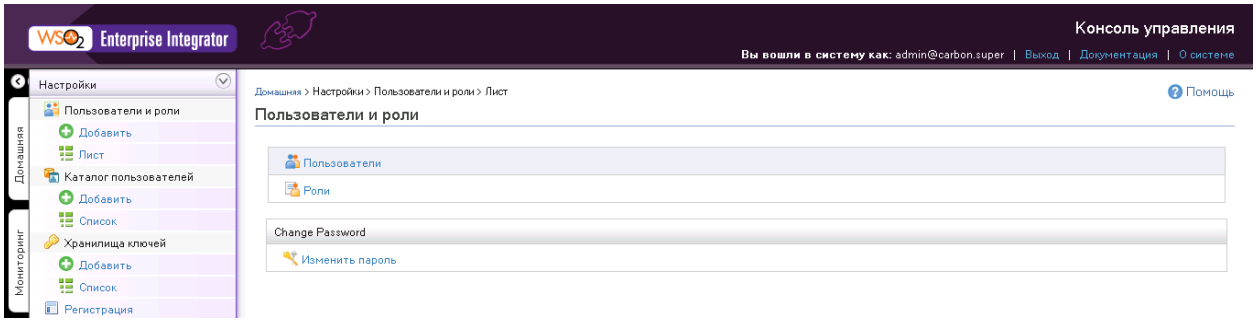


Рисунок 57. Графический интерфейс пункта «Список».

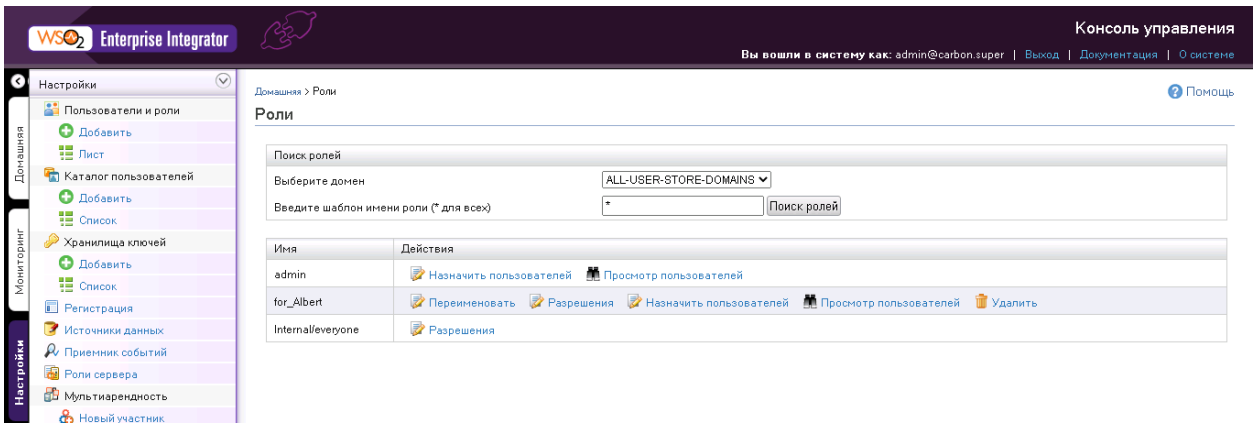


Рисунок 58. Графический интерфейс пункта «Список», пункт «Пользователи».

5.3.2 Раздел «Пользовательские хранилища»

Хранилище пользователей - это база данных, в которой хранится информация о пользователях и ролях пользователей, включая логин, пароль, Ф.И.О. и адрес электронной почты.

Функция управления пользователями предоставляется по умолчанию во всех продуктах WSO2 и настраивается в user-mgt.xml файле, который находится в <EI_HOME>/repository/conf/каталоге.

По умолчанию встроенная база данных H2 (JDBC), настроена в качестве основного хранилища пользователей, за исключением сервера идентификации WSO2. Вы можете изменить настройки по умолчанию, заменив базу данных в соответствии с требованиями. Основное хранилище пользователей является общим для всех пользователей в системе.

Можно настроить любое из следующих типов пользовательских хранилищ:

- Хранилище пользователей JDBC;
- Хранилища пользователей Active Directory;
- Хранилища пользователей LDAP - только для чтения;
- Чтение-запись пользовательских хранилищ LDAP.

С помощью функции управления пользователями любое количество вторичных хранилищ пользователей может быть легко настроено для вашей системы с помощью консоли управления. Это автоматически создаст XML-файл с конфигурациями, соответствующими вторичному пользовательскому хранилищу в том же <EI_HOME>/repository/conf/каталоге. Кроме того, можно вручную создать файл конфигурации и сохранить его в этом каталоге без использования консоли управления.

Для перехода в раздел «**Пользовательские хранилища**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Настройки**» > «**Пользовательские хранилища**»

5.3.2.1 «Добавить» (

Для перехода в интерфейс пункта «**Добавить**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Настройки**» > «**Пользовательские хранилища**» > «**Добавить**»

Система отобразит графический интерфейс с набором элементов, которые позволяют добавить нового менеджера для хранилища (вторичное) пользователей (см. Рисунок 59).

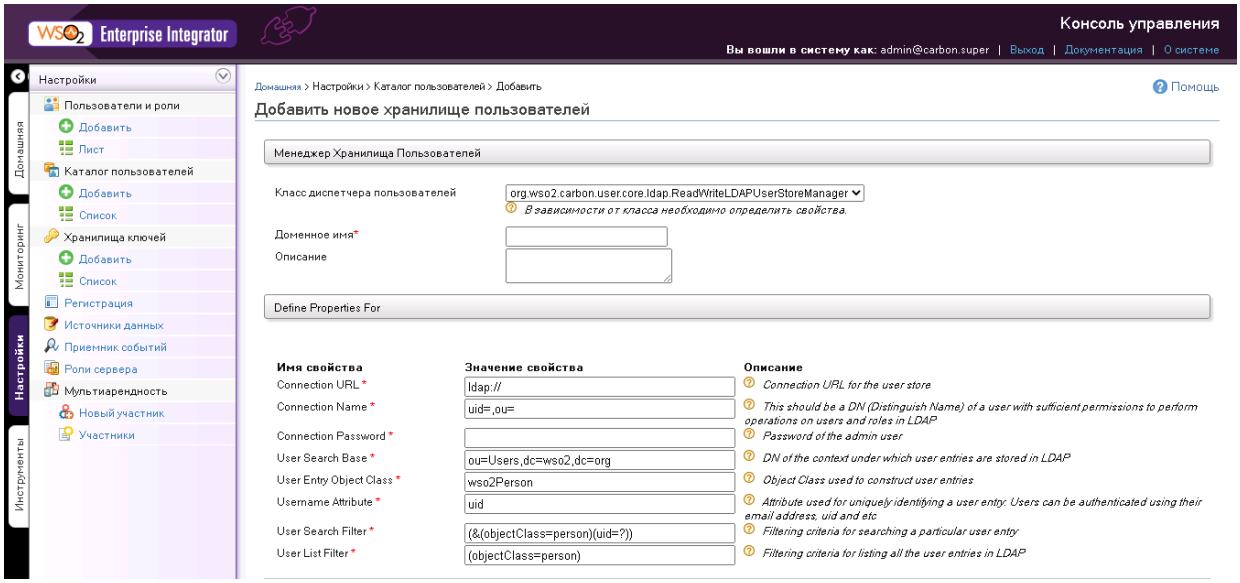


Рисунок 59. Графический интерфейс пункта «Добавить».

Создание вторичного хранилища, автоматически сгенерирует XML-файл с заданной конфигурацией в <EI_HOME>/repository/conf/каталог.

5.3.2.2 «Список»

Для перехода в интерфейс пункта «**Список**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Настройки**» > «**Пользовательские хранилища**» > «**Список**»

Система отобразит графический интерфейс с набором элементов, которые позволяют получить информацию о хранилищах пользователей и осуществить управление ими (см. Рисунок 60).

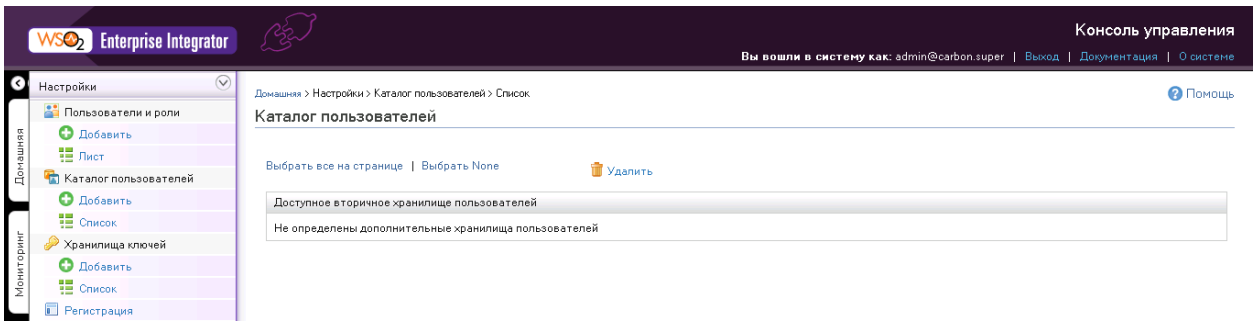


Рисунок 60. Графический интерфейс пункта «Список».

5.3.3 Раздел «Хранилище ключей»

Хранилище ключей - это хранилище, в котором хранятся криптографические ключи и сертификаты, используемые для обеспечения безопасности, таких как шифрование конфиденциальной информации и установление защищенного/зашифрованного соединения между сервером и внешними сторонами, которые подключаются к вашему серверу.

По умолчанию, хранилище ключей wso2carbon.jks хранится в <EI_HOME>/repository/resources/security каталоге. Это хранилище ключей поставляется с парой закрытых / открытых ключей, которая используется для шифрования конфиденциальной информации, для связи по протоколу SSL и для целей шифрования/подписи в WS-Security.

Следует обратить внимание - любой пользователь WSO2 имеет доступ к предустановленному хранилищу ключей, так как WSO2 это продукт с открытым исходным кодом. Рекомендуется заменить имеющееся предустановленное хранилищем ключей, когда продукты развертываются в производственных средах.

Пары ключей: согласно криптографии, с открытым ключом, пара ключей (закрытый ключ и соответствующий открытый ключ) используется для шифрования конфиденциальной информации и для аутентификации личности сторон, которые взаимодействуют с вашим сервером.

Цифровой сертификат: при наличии пары ключей, необходимо иметь цифровой сертификат для проверки подлинности ключей. Как правило, открытый ключ пары ключей встроен в цифровой сертификат, который также содержит дополнительную информацию, такую как владелец, срок действия и т.д.

Доверенные сертификаты: для установления доверия цифровой сертификат, содержащий открытый ключ, должен быть подписан доверенным Удостоверяющим центром (CA). Вы можете создать самозаверяющие сертификаты для открытого ключа (тем самым создав свой собственный центр сертификации) или получить сертификаты, подписанные внешним центром сертификации.

Оба типа доверенных сертификатов могут эффективно использоваться в зависимости от информации, защищаемой ключами. Сертификат, подписанный надежным и доверенным центром сертификации, обеспечивает высокий уровень доверия у сторон, использующих его.

Для перехода в раздел «**Хранилище ключей**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Настройки**» > «**Хранилище ключей**»

5.3.3.1 «Добавить»

Для перехода в интерфейс пункта «**Добавить**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Настройки**» > «**Хранилище ключей**» > «**Добавить**»

Система отобразит графический интерфейс с набором элементов, которые позволяют добавить хранилище ключей (см. Рисунок 61).

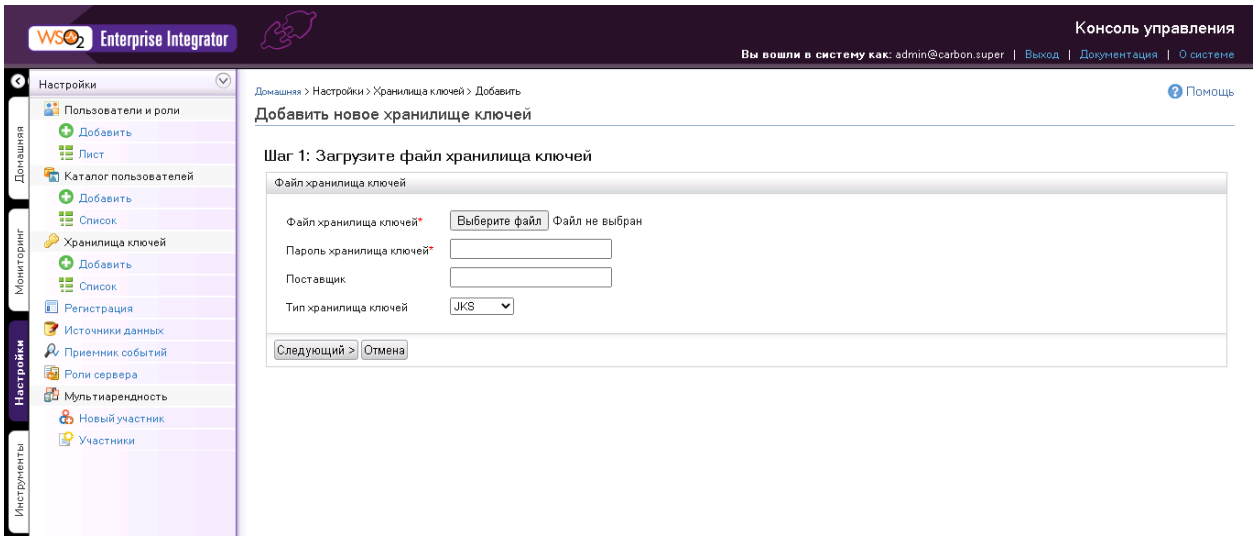


Рисунок 61. Графический интерфейс пункта «Добавить».

5.3.3.2 «Список»

Для перехода в интерфейс пункта «**Список**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Настройки**» > «**Хранилище ключей**» > «**Список**»

Система отобразит графический интерфейс с перечнем хранилищ ключей и набором элементов, которые позволяют управлять ими (см. Рисунок 62).

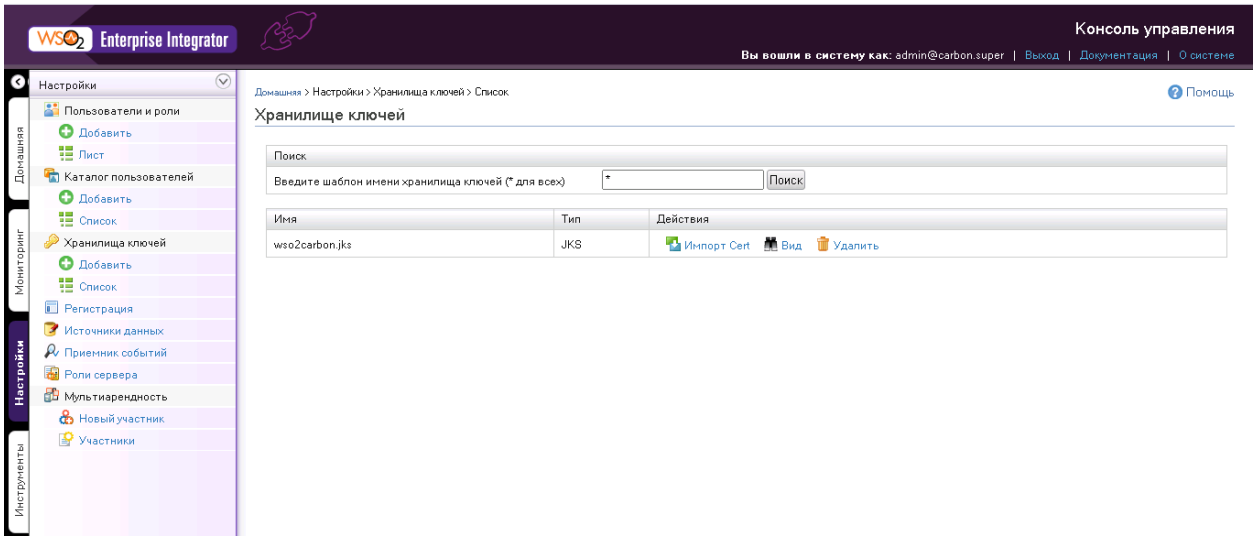


Рисунок 62. Графический интерфейс пункта «Список».

5.3.4 «Логирование»

Продукты WSO2 обладают возможностью ведения журнала (логирование) и записывать события, происходящие на стороне сервера.

При изменении параметров с помощью консоли управления, сначала сервер сохраняет новые значения в базе данных, а затем изменяет соответствующие компоненты в структуре ведения журнала, позволяя быстро обновить настройки ведения журнала.

В дополнение к журналам log4j2 существует возможность включить журналы HTTP-доступа для отслеживания информации, как лица, получающие доступ к профилю EI, количество полученных обращений, количество ошибок и т. д.

Для перехода в интерфейс пункта «**Логирование**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Настройки**» > «**Логирование**»

Система отобразит графический интерфейс набором элементов, которые позволяют настраивать правила логирования для шины, отдельно взятого приложения и логера (см. Рисунок 63).

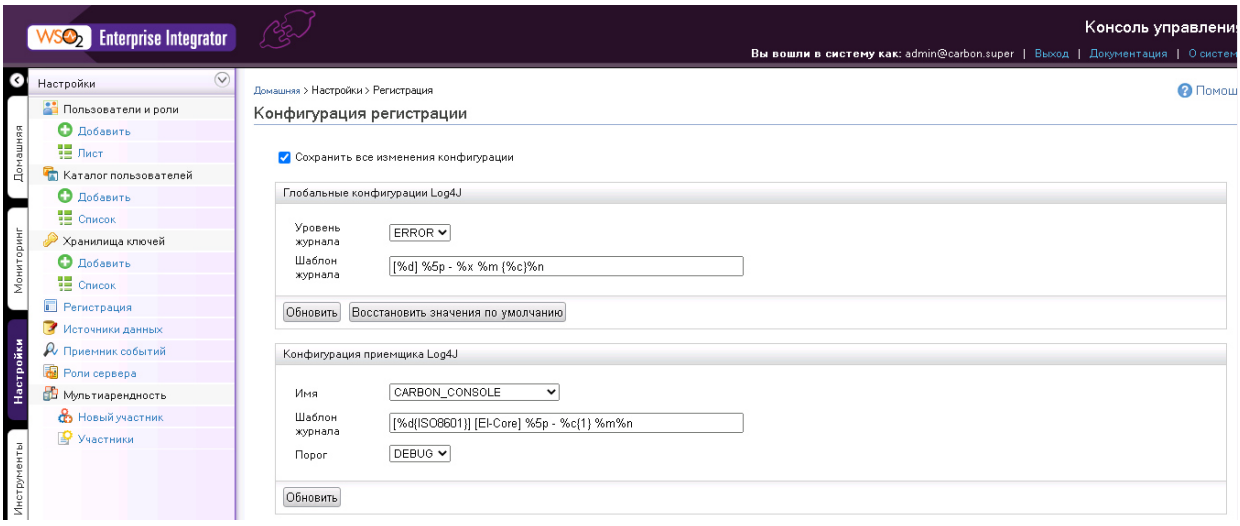


Рисунок 63. Графический интерфейс пункта «Логирование».

Уровень логирования порогового значения, может быть установлен специально для каждого приложения в системе. Если уровень порогового значения специально не задан для приложения, то предустановленный уровень логирования по умолчанию, будет применяться ко всем приложениям.

Уровни журнала, которые можно настроить:

Уровень	Описание
OFF (выкл)	Отключение ведения журнала.
FATAL (фатальный)	Ошибки сервера, которые приводят к преждевременному завершению работы.
ERROR (ошибка)	Ошибки во время выполнения или непредвиденные условия. Эти логи будут сразу же видны в командной строке, используемой для запуска сервера.
WARN (предупреждение)	Указывает на использование устаревших API, некорректное использование API, и другие ошибки которые являются нежелательными или неожиданными, но не критичными. Эти логи будут сразу же видны в командной строке, используемой для запуска сервера.
INFO (информация)	Указывает на важные события среды выполнения, такие как запуск/завершение работы сервера. Эти логи будут сразу же видны в командной строке, используемой для запуска сервера. Рекомендуется свести ведение этих журналов к минимуму.
DEBUG (отслеживать)	Предоставляет подробную информацию о информационном потоке проходящим через систему. Эта информация будет записываться только в журналы. Большинство логов, регистрируемых вашим приложением, должны быть записаны в виде журнала отладки.
TRACE (след)	Предоставляет дополнительные сведения о поведении событий и служб. Предполагается, Эта информация будет записываться только в журналы.

5.3.5 «Источники данных»

Источники данных предоставляют информацию, которую сервер может использовать для подключения к базе данных.

Можно просматривать, редактировать и удалять источники данных в консоли управления, однако нельзя редактировать или удалять следующие источники данных:

- REGISTRY_LOCAL1
- REGISTRY_LOCAL2
- REGISTRY_DB
- WSO2_USER_DB
- PERSISTENCE_DB
- WSO2_CARBON_DB
- EI_ANALYTICS
- PERSISTENCE_DB

Изменение вышеперечисленных источников данных производится в файле:

<EI_HOME>/repository/conf/datasources/master-datasources.xml

Для перехода в интерфейс пункта «**Источники данных**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Настройки**» > «**Источники данных**»

Система отобразит графический интерфейс с перечнем источников данных и набором элементов, которые позволяют добавить новый источник данных или управлять имеющимися (см. Рисунок 64).

Для добавления источника данных, выполните следующие действия:

- Нажать “Добавить источник данных”.
- Указать необходимые параметры для подключения к базе данных. Доступные параметры зависят от типа источника данных, которые создаются.

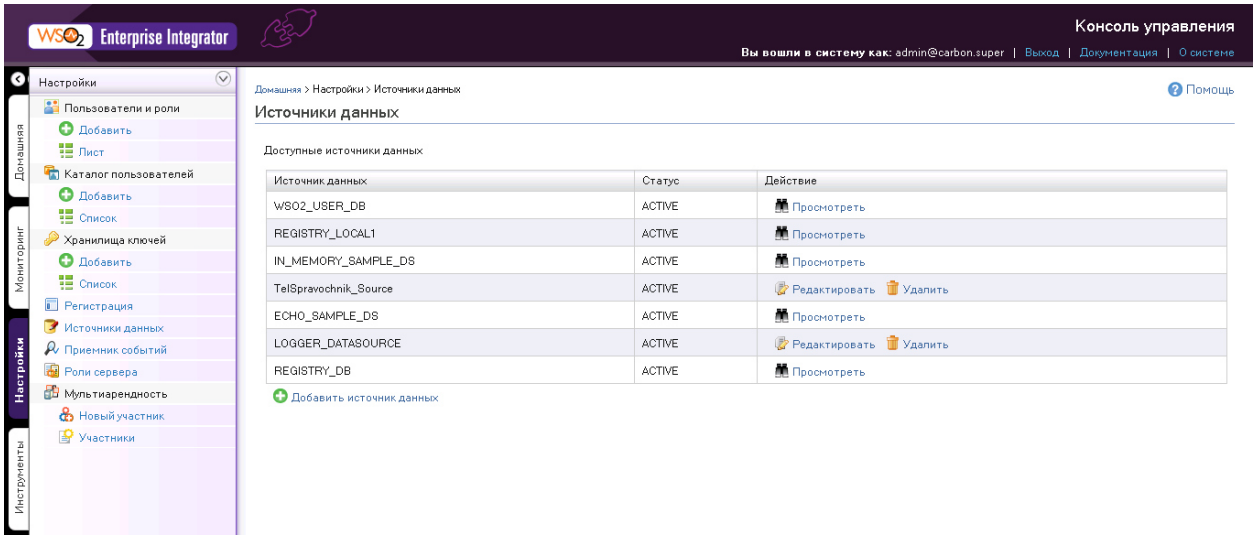


Рисунок 64. Графический интерфейс пункта «Источники данных».

Для создания нового источника данных, необходимо ввести значения следующих полей:

- Тип источника данных.
- Name (Имя): имя источника данных (должно быть уникальным значением)
- Поставщик источника данных: укажите поставщика источника данных.
- Driver: имя класса используемого драйвера.
- URL: URL-адрес для установления соединения.
- Имя пользователя: имя пользователя соединения, которое использовано для установления соединения.
- Пароль: Пароль соединения, который будет использован для установления соединения.

Перед тем как сохранить источник данных, можно проверить соединение. Если соединение прошло успешно. Нажмите кнопку сохранить.

Важно: Драйвер JDBC должен быть установлен в системе

При необходимости и наличии драйвера в системе, можно использовать подключения к другим СУБД.

5.3.6 «Приемник событий»

Приемники событий содержат информацию о конечных точках транспорта, предоставляемых другими системами. События могут быть размещены в этих конечных точках. URL-адреса транспорта конечных точек, данные аутентификации и учетные данные регистрируются в приемниках событий.

Для перехода в интерфейс пункта «**Приемник событий**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Настройки**» > «**Приемник событий**»

Система отобразит графический интерфейс с набором элементов, которые позволяют добавить новый приемник событий (см. Рисунок 65).

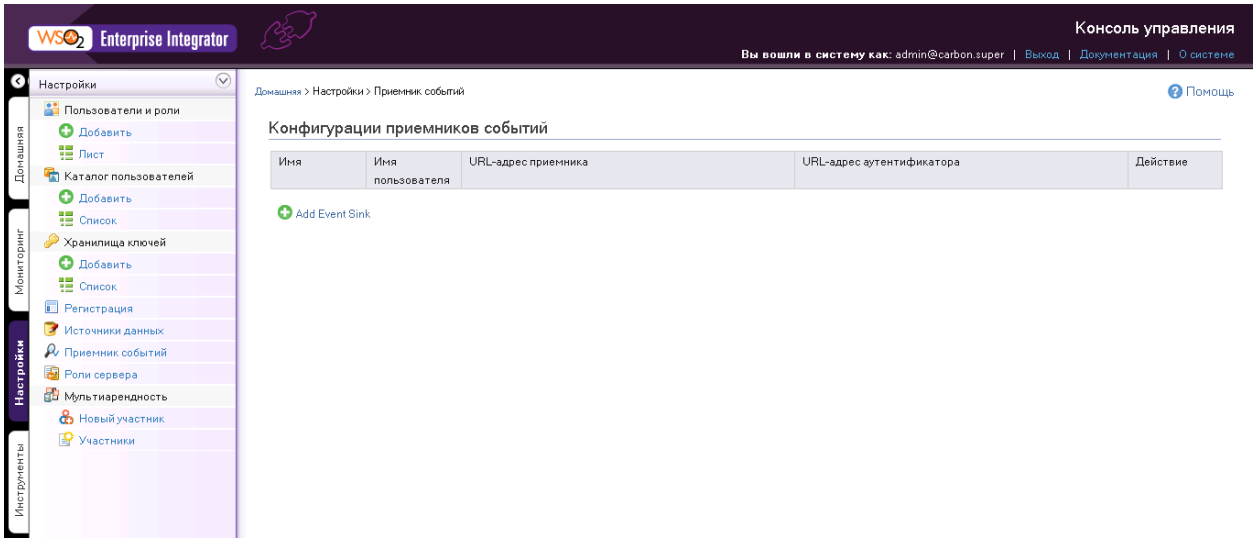


Рисунок 65. Графический интерфейс пункта «Event Sinks».

5.3.7 «Роли сервера»

Роль сервера - это параметр, который упоминается в carbon.xml для продуктов на основе wso2carbon. Каждый продукт имеет определенное свойство ServerRoles по умолчанию в файле carbon.xml. Значения ролей сервера по умолчанию для продуктов на основе WSO2Carbon следующие:

- WSO2 AS - "ApplicationServer";
- WSO2 BAM - "BusinessActivityMonitor";
- WSO2 BPS - "BusinessProcessServer";
- WSO2 BRS - "BusinessRulesServer";
- WSO2 DSS - "DataСервисыServer";
- WSO2 ESB - "EnterpriseServiceBus";
- WSO2 GS - "GadgetServer";
- WSO2 GReg - "GovernanceRegistry";
- WSO2 IS - "IdentityServer";
- WSO2 MS - "MashupServer".

Этот параметр используется при развертывании архивов C-App. Как правило, C-App состоит из набора ресурсов, которые должны быть развернуты в продукте. Для сортировки того, что может быть развернуто, а что нет, используется свойство роли сервера. Когда происходит развертывание C-App, считывается свойство ServerRoles в carbon.xml и развертываются только те ресурсы, которые соответствуют значениям роли сервера в нем.

Список ресурсов C-App, которые сопоставляются с ролями сервера по умолчанию.

- ApplicationServer - foo.aar$
- EnterpriseServiceBus - proxy.xml;
- BusinessProcessServer - my_bpel.zip;
- GadgetServer - gadget.xml;
- Роли Сервера Управлятьr.

Диспетчер ролей сервера хранить два файла carbon.xml-роли продукта по умолчанию и роли сервера, указанные пользователем/арендатором, в реестре конфигурации. Поэтому, когда C-App развертывается в carbon, средство развертывания C-App будет проверять наличие автоматически упомянутых ролей сервера из реестра вместо carbon.xml.

В диспетчере ролей сервера свойства ролей сервера имеют два типа:

Default (По умолчанию) - все роли сервера, выбранные из специального экземпляра carbon.xml.

Custom - все остальные роли сервера, добавленные пользователями.

Для перехода в интерфейс пункта «**Роли Сервера**», выполните следующие действия:

Выберите вкладку «**Домашняя**» > «**Настройки**» > «**Роли Сервера**»

Система отобразит графический интерфейс с перечнем имеющихся серверных ролей и набором элементов, которые позволяют добавить новую серверную роль (см. Рисунок 66).

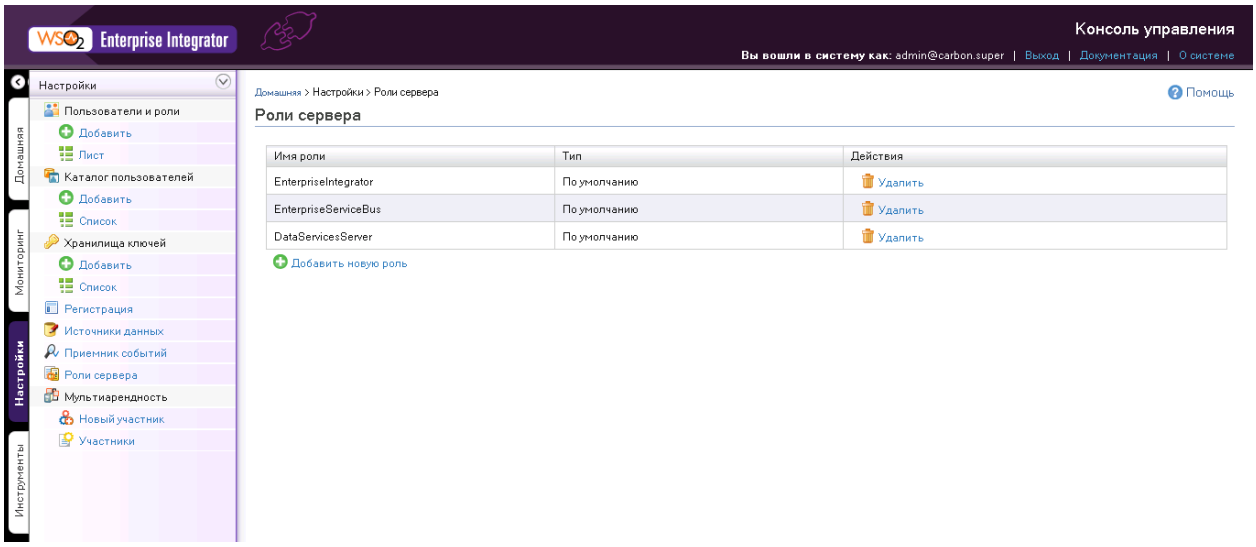


Рисунок 66. Графический интерфейс пункта «Роли Сервера».

6 Администрирование API Manager

6.1 Управление пользователями и ролями

6.1.1 Настройка хранилища пользователей. Подключение к Active Directory

Подключитесь к интерфейсу администрирования API Manager https://apim-domain-name/carbon. Выберите пункт меню Main / User Stores / Add. Заполните следующие параметры:

- User store manager class - класс обработчика, выберите org.wso2.carbon.user.core.ReadOnlyLDAPUserStoreManager для работы с Active Directory с правами только для чтения;
- Domain name - любое уникальное имя для хранилища;
- Connection URL - по схеме ldap://<адрес контроллера AD>:389; порт указывать обязательно;
- Connection name - имя пользователя для подключения к контроллеру, любой пользователь с правами чтения каталога AD;
- User search base - уникальное имя (DN, distinguished name) раздела LDAP, в котором производить поиск учетных записей; рекомендуем задавать максимально узкий диапазон для уменьшения нагрузки и ускорения работы;
- Username attribute - имя атрибута LDAP объекта, в котором хранится имя пользователя; для AD - sAMAccountName;
- User search filter - LDAP фильтр для поиска пользователя; для AD - (&(objectClass=person)(sAMAccountName=?));
- User list filter - LDAP фильтр для поиска учетных записей пользователей; (objectClass=person);
- Read groups - загружать ли из AD информацию о группах; так как управление ролями/группами будет происходить на стороне API Manager, отключите эту опцию.

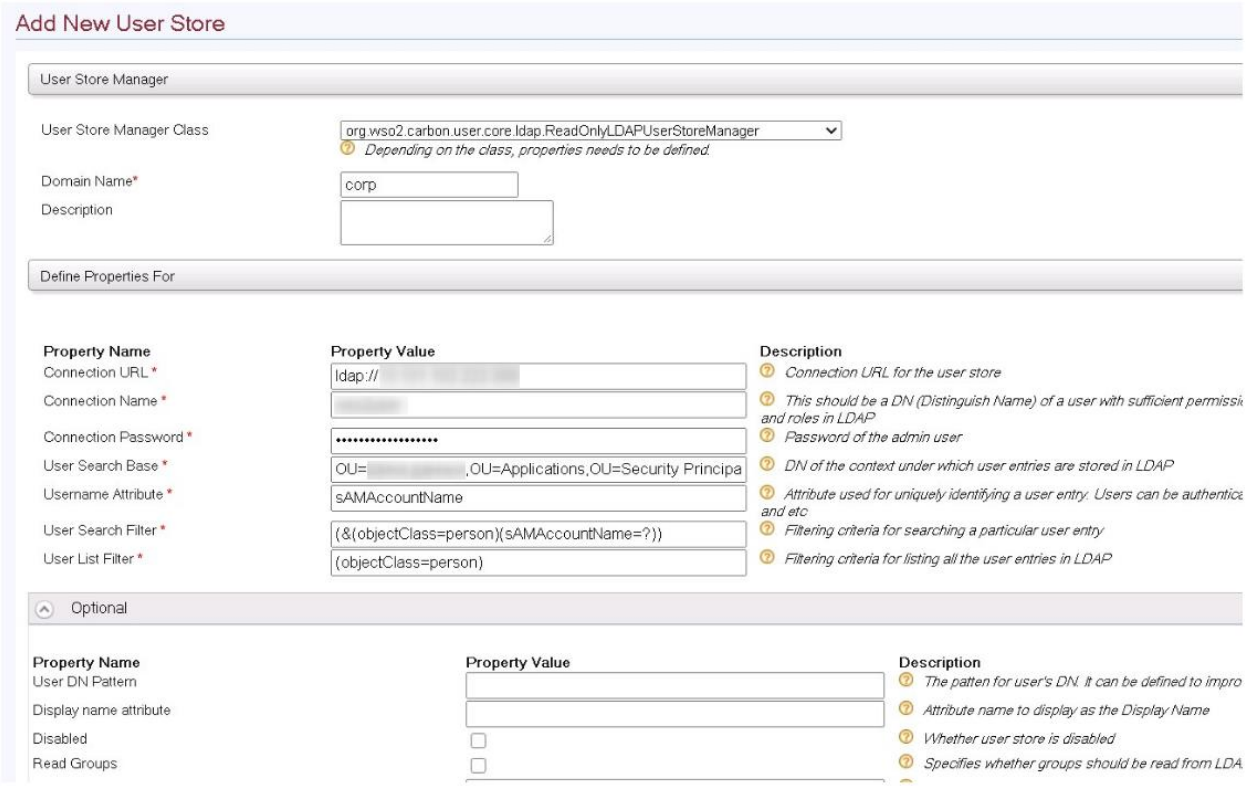


Рисунок 6 Параметры хранилища учетных записей

Нажмите кнопку Add для подключения хранилища. Хранилище подключится в течение нескольких секунд. Обновите экран. После подключения страница должна иметь следующий вид.

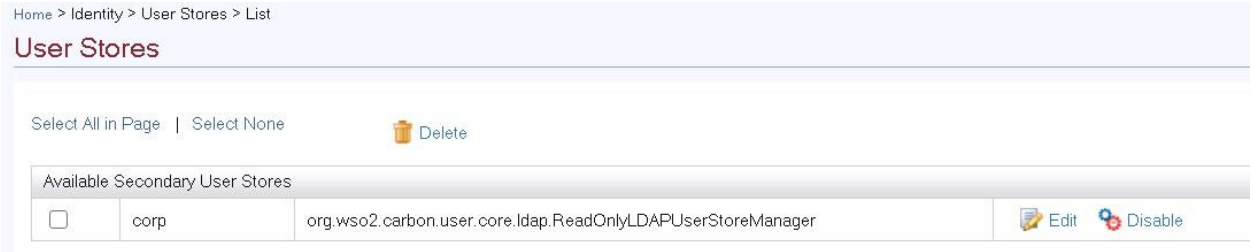


Рисунок 7 Список хранилищ учетных записей

Перейдите в раздел Users and Roles / List / Users. В списке должны отобразиться пользователи из AD. Префикс таких пользователей - название хранилища (Domain name).

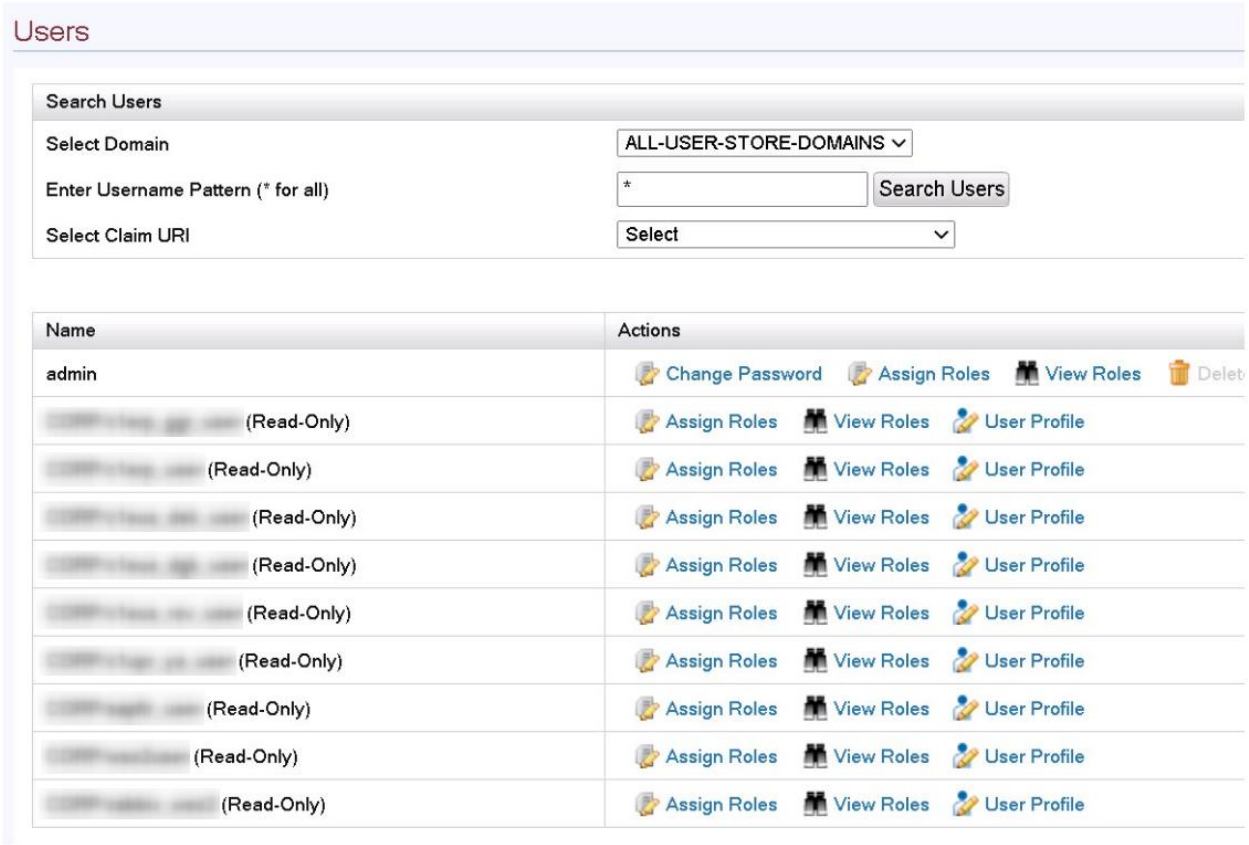


Рисунок 8 Список пользователей

6.1.2 Добавление роли

Перейдите в раздел Users and Roles / Add / Add new role. Выберите домен INTERNAL (только в роли из этого домена можно добавлять пользователей из внешнего хранилища). Введите имя роли. Нажмите Next.

На следующем шаге выбора разрешений установите Login: права аутентифицироваться в системе. Нажмите Next.

Выберите пользователей, добавляемых к роли. Нажмите Finish.

Для добавления пользователей к существующей роли перейдите в раздел Users and Roles / List / Roles и нажмите Assign users напротив роли, отметьте пользователей и нажмите Finish.

Для того, чтобы пользователи смогли авторизоваться при запросах к API добавьте их к встроенной роли Internal/subscriber.

6.2 Создание областей действия API

Области действия (scope) используются для удобства назначения прав использования API.

Откройте интерфейс разработчика API https://apim-domain-name/publisher. Перейдите в раздел Scopes

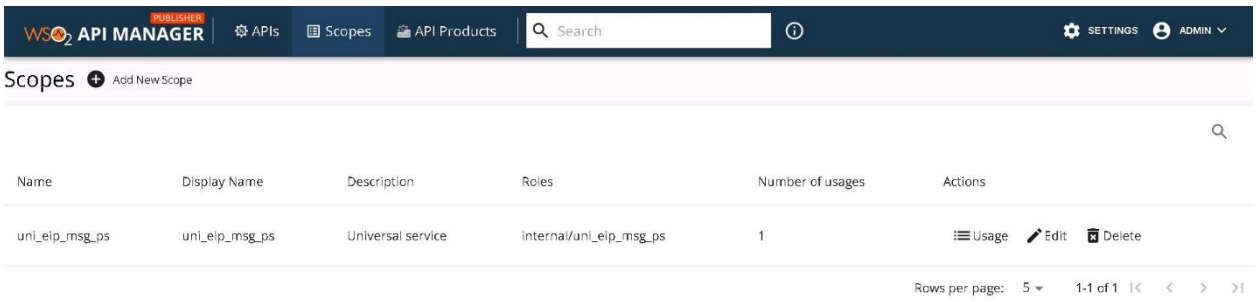


Рисунок 9 Области действия

Нажмите Add new scope и заполните поля:

- Name - наименование области;
- Display name - отображаемое имя;
- Description - описание;
- Roles - роли, которым дается права использования ресурсов из данной области; введите полное название существующей роли (см. п. 6.1.2), например «internal/uni_eip_msg_ps» и нажмите Enter; после проверки существования данной роли наименование отобразится в виде виджета, изображенного на рисунке Рисунок 10 Создание области действия.

Нажмите Save.

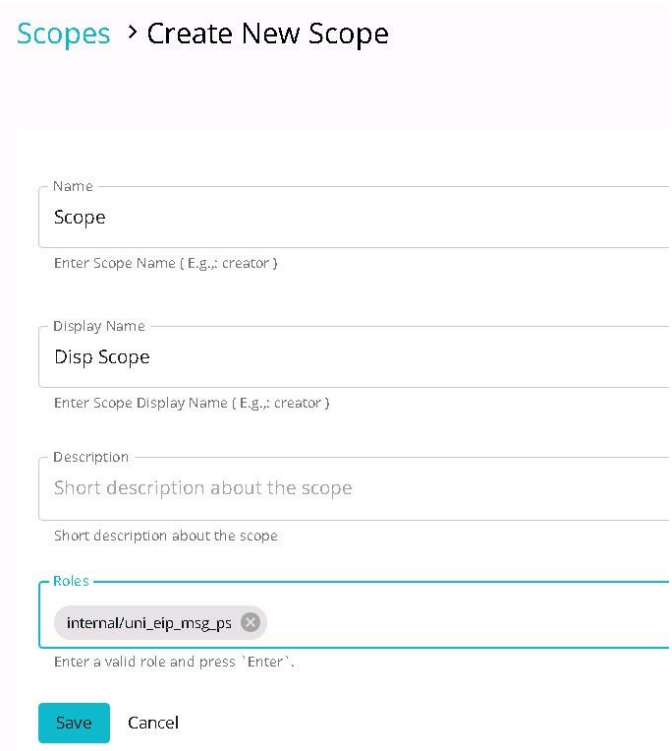


Рисунок 10 Создание области действия

6.3 Создание API

6.3.1 Подготовка WSDL-описания для API

Откройте интерфейс администрирования WSO2 EI https://ei-domain-name:9443/carbon. Перейдите в раздел сервисов и найдите прокси-сервис ИП. Сохраните на диск WSDL описание сервиса.

Если в описании сервиса импортируется XSD схема данных, сохраните ее на диск. Для этого к адресу сервиса добавьте параметр xsd, из тега schemaLocation.

Например, адрес сервиса Универсальный интерфейс http://ei-domain-name/services/uni_eip_msg_ps. URL WSDL описания http://ei-domain-name/services/uni_eip_msg_ps?wsdl. В описание импортируется схема данных следующим образом

```
      <xsd:schema attributeFormDefault="unqualified"
elementFormDefault="unqualified">
      <xsd:import namespace="http://www.rushydro.ru/eip/msg"
schemaLocation="uni_eip_msg_ps?xsd=uni_eip_msg_xsd.xsd"/>
      </xsd:schema>
```

URL схемы данных, в данном случае: http://ei-domain-name/services/uni_eip_msg_ps?xsd=uni_eip_msg_xsd.xsd

Отредактируйте WSDL описание: замените вышеуказанный тег с импортом схемы тегом <xsd:schema> вместе с вложенными элементами из файла со схемой. Данная операция нужна, так как API Manager не предоставляет возможности размещения произвольных ресурсов (схемы данных) и автоматической подстановки в описание.

Альтернативный вышеописанному редактированию вариант: заменить значение атрибута schemaLocation на полный URL к схеме данных на сервере ESB http://ei-domain-name/services/uni_eip_msg_ps?xsd=uni_eip_msg_xsd.xsd

6.3.2 Создание API

Откройте интерфейс разработчика API https://apim-domain-name/publisher. Перейдите в раздел APIs.

Нажмите Create API и выберите «I have a SOAP endpoint». На следующей странице выберите Implementation type - Pass through (прозрачный); Input type - WSDL file/archive. Перетащите на панель виджета файл, подготовленный в предыдущем разделе, или нажмите Browse file, найдите файл и загрузите. Нажмите Next.

Заполните поля и Нажмите Create:

- Name - название API;
- Context - контекст/путь URL; укажите название сервиса ИП (в соответствии с «Регламент проектирования интеграционных задач»);
- Version - номер версии API; укажите в формате «v<номер>»;
- Endpoint - URL WSDL сервиса ИП на основном узле ESB (http://ei-domain-name:8280/services/<имя сервиса>?wsdl).

7 Администрирование Apache Kafka

7.1 Администрирование Apache Kafka

В Руководстве представлены типичные команды и их параметры. Подробную информацию смотрите в официальной документации Apache Kafka.

7.1.1 Настройка и установка сертификатов

Сертификаты SSL/TLS используются для взаимной аутентификации брокеров, а также для установления защищенного канала между брокером и клиентом.

Сгенерируйте ключ и сертификат для сервера.

```
keytool -keystore kafka-domain-name.jks -storetype pkcs12 -alias localhost -
validity 3650 -genkey -keyalg rsa -keysize 2048 -dname CN=kafka-domain-
name,L=Moscow,C=RU,O='PJSC RusHydro' -ext SAN=DNS:kafka-domain-name,IP:kafka-
ip -ext EKU=clientAuth,serverAuth -keypasswd pass -storepasswd pass
```

Пароли к закрытому ключу и к хранилищу - произвольные, укажите их в конфигурации (см. далее).

При формировании сертификата используйте в параметре CN полное доменное имя узла, например: kafka-domain-name. По этому полю происходит определение имени пользователя при аутентификации брокеров, как указано в настройках сервера.

Сформируйте запросы на подпись сертификатов и подпишите их в Удостоверяющем Центре.

При оформлении заявки на подпись укажите, что необходимо подписывать с указанием параметра **extendedKeyUsage=clientAuth,serverAuth** (для взаимной аутентификации).

```
keytool -keystore kafka-domain-name.jks -storetype pkcs12 -alias localhost -
certreq  -ext SAN=DNS:kafka-domain-name,IP:kafka-ip -ext
EKU=clientAuth,serverAuth -file kafka-domain-name.csr
```

Импортируйте подписанные сертификаты в хранилища. При этом необходимо импортировать корневой и все промежуточные сертификаты Удостоверяющего Центра. В инструкции: корневой сертификат расположен в файле rootca.pem; промежуточный-issuerca.pem, подписанный сертификат – kafka-domain-name_signed.pem

```
keytool -keystore kafka-domain-name.jks -import -file rootca.pem -alias
rootca

keytool -keystore kafka-domain-name.jks -import -file issuerca.pem -alias
issuerca

keytool -keystore kafka-domain-name.jks -import -file kafka-domain-
name_signed.pem -alias localhost
```

Полученный файл хранилища скопируйте на сервер, пропишите путь и остальные параметры в конфигурации брокера Kafka.

Поместите сертификаты удостоверяющего центра в хранилище доверенных сертификатов

```
keytool -keystore kafka.client.truststore.jks -import -file rootca.pem -alias
rootca -storetype pkcs12 -storepasswd pass
keytool -keystore kafka.client.truststore.jks -import -file issuerca.pem -
```

```
alias issuerca
```

Полученный файл скопируйте на все узлы Kafka, укажите параметры в конфигурации брокера.

7.1.2 Аутентификация Zookeeper

Включите аутентификацию клиентов на сервере zookeeper. добавьте в файл конфигурации zookeeper.properties

```
authProvider.1=org.apache.zookeeper.server.auth.SASLAuthenticationProvider
```

Чтобы обязать всех клиентов использовать аутентификацию, добавьте параметр

```
requireClientAuthScheme=sasl
```

Создайте файл /opt/kafka/config/zookeeper_jaas.conf с настройками пользователей и паролей самого сервера zookeeper и брокера kafka, в качестве клиента.

```
      Server {
             org.apache.zookeeper.server.auth.DigestLoginModule required
             user_super="пароль zookeeper"
             user_kafka="пароль брокера кафки";
      };
```

7.1.3 Аутентификация Kafka

Отредактируйте файл /opt/kafka/server.properties

Параметры

```
      listeners=SSL://:9093,SASL_SSL://:9094
      sasl.enabled.mechanisms=PLAIN
      security.inter.broker.protocol=SSL
      ssl.client.auth=required
```

устанавливают механизм взаимной аутентификации SSL/TLS между брокерами на порту 9093, и механизма SASL (с нешифрованным паролем) поверх SSL/TLS туннеля для клиентов на 9094 порту.

Параметры

```
      ssl.truststore.location=/opt/kafka/config/security/kafka.client.truststo
re.jks
      ssl.truststore.password=pass
      ssl.truststore.type=pkcs12
      ssl.keystore.location= /opt/kafka/config/security/kafka-domain-name.jks
      ssl.keystore.password=pass
      ssl.key.password=pass
      ssl.keystore.type=pkcs12
```

настраивают хранилище собственного сертификата сервера (keystore) и хранилище доверенных сертификатов клиентов (корневой и промежуточные сертификаты Удостоверяющего центра)

Параметр

```
      authorizer.class.name=kafka.security.authorizer.AclAuthorizer
```

указывает java-класс для обработки авторизации по спискам доступа (ACL)

Параметры

```
      super.users=User:CN=kafka-domain-name;User:CN=kafka-domain-
name;User:admin

      ssl.principal.mapping.rules=\
      RULE:^CN=(.*?),/CN=$1/,\
      RULE:^CN=(.*?),O=(.*?),L=(.*?),C=(.*?)$/CN=$1/,\
      RULE:^(CN=.*?),OU=(.*?),O=(.*?),L=(.*?),ST=(.*?),C=(.*?)$/$1/,\
      DEFAULT
```

настраивают пользователей с правами администратора “admin” и с именами извлекаемыми из сертификатов. Также, указывают правила извлечения имени из сертификата с использованием регулярных выражений.

Файл /opt/kafka/config/kafka_jaas.conf служит для настройки доступа брокера kafka к серверу zookeeper (секция Client) и указание имен пользователей и паролей для клиентов брокера и способа аутентификации (через указание java-класса). Имена клиентов задаются в формате user_<имя пользователя>.

```
      Client {
        org.apache.zookeeper.server.auth.DigestLoginModule required
        username="kafka"
        password="пароль брокера кафки";
      };
      KafkaServer {
         org.apache.kafka.common.security.plain.PlainLoginModule required
         username="admin"
         password="пароль администратора"
         user_admin=" пароль администратора "
         user_eip="пароль пользователя для ЕИП"
         user_ek="пароль пользователя для АИС Единого Казначейства";
      };
```

Для аутентификации на брокерах Kafka от имени пользователя admin при запуске консольных утилит создайте файл /opt/kafka/config/admin_client.properties

```
      security.protocol=SASL_SSL
      ssl.truststore.location=/opt/kafka/config/security/kafka.client.truststo
re.jks
      ssl.truststore.password=pass
      sasl.mechanism=PLAIN
      sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule
required \
          username=\"admin\" \
          password=\"пароль администратора\";
```

7.2 Управление топиками.

7.2.1 Создание топика

Пример команды:

```
      bin/kafka-topics.sh --bootstrap-server kafka-domain-name:9094 --command-
config config/security/admin_client.properties --create --partitions 1 --
replication-factor 2 --topic eip_logger
```

replication-factor - количество реплик (резервных копий) д.б. не больше количества брокеров в кластере

partitions - количество разделов (партиций) в топике; используется, например, для параллелизации обработки сообщений. Количество потребителей в одной группе (consumer group), получающих сообщения из топика, не будет больше количества партиций. Т.е. если необходимо обрабатывать сообщения двум потребителям, то необходимо создавать топик с не менее чем 2-мя партициями. Подробнее читайте в официальной документации Kafka.

7.2.2 Настройка доступа (ACL)

Разрешение всем пользователям использовать транзакции kafka

```
bin/kafka-acls.sh --bootstrap-server kafka-domain-name:9094 --command-config
config/security/admin_client.properties \
 --add --allow-principal User:* --transactional-id '*' --operation describe -
-operation write
```

Разрешение пользователю ek подключаться с указнием групп потребителей (consumer group) имена, которых начинаются на «ek-group»

```
bin/kafka-acls.sh --bootstrap-server kafka-domain-name:9094 --command-config
config/security/admin_client.properties \
 --add --allow-principal User:ek --group ek-group --resource-pattern-type
prefixed
```

Разрешение поьльзовтелю ek писать, читать и получать информацию о топиках AIS_2LOCALS и LOCALS_2AIS

```
bin/kafka-acls.sh --bootstrap-server kafka-domain-name:9094 --command-config
config/security/admin_client.properties \
 --add --allow-principal User:ek --operation Write --operation Describe --
operation Read --topic AIS_2LOCALS --topic LOCALS_2AIS
```

Просмотр текущих установок ACL

```
bin/kafka-acls.sh --bootstrap-server kafka-domain-name:9094 --command-config
config/security/admin_client.properties --list
```

7.3 Основные операции с топиками

Просмотр сообщений в топике с самого старого сообщения (приведены 2 варианта вызова)

```
bin/kafka-console-consumer.sh --bootstrap-server kafka-domain-name --
consumer.config config/security/client_security.properties  --topic
AIS_2LOCALS --group eip-group-test --from-beginning --property
print.timestamp=true

bin/kafka-console-consumer.sh --bootstrap-server kafka-domain-name:9094 --
consumer.config config/security/client_security.properties  --topic
AIS_2LOCALS --partition 0 --offset earliest --property print.timestamp=true
```

Список всех групп потребителей

```
bin/kafka-consumer-groups.sh --bootstrap-server kafka-domain-name:9094 --
```

```
command-config config/security/admin_client.properties --list
```

Вывести информацию о группе потребителей

```
bin/kafka-consumer-groups.sh --bootstrap-server kafka-domain-name:9094 --
command-config config/security/admin_client.properties --group eip-group-
uni_adp --describe
```

Команда выведет примерный результат

```
GROUP            TOPIC          PARTITION  CURRENT-OFFSET  LOG-END-OFFSET  LAG
CONSUMER-ID                                    HOST
CLIENT-ID
eip-group-uni_adp uni_req_2_eip   0          35079           35079           0
consumer-2-909fd3ce-d58b-4ecf-b3d9-b2a89742fa90 kafka-domain-name/kafka-ip consumer-2
```

Здесь в столбце topic перечислены топики, к которым данная группа подключалась;

Partition - раздел топика;

LOG-END-OFFSET - номер последнего сообщения в топике;

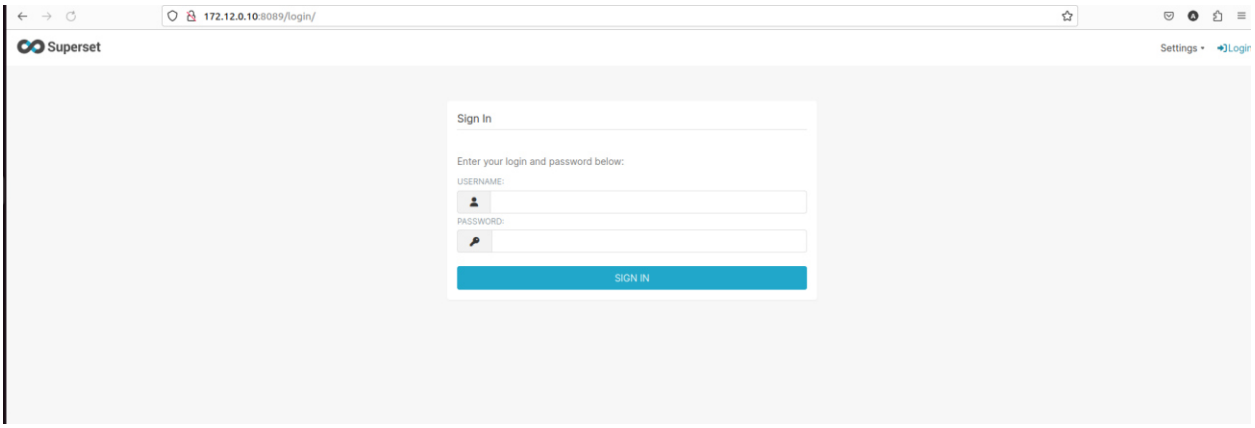
CURRENT-OFFSET - номер последнего сообщения, прочитанного потребителями группы;

LAG - отставание (разница между вышеописанными значениями);

CONSUMER-ID, HOST, CLIENT-ID - сведения о текущих подключенных потребителях. Если нет подключений в столбцах печатается «-».

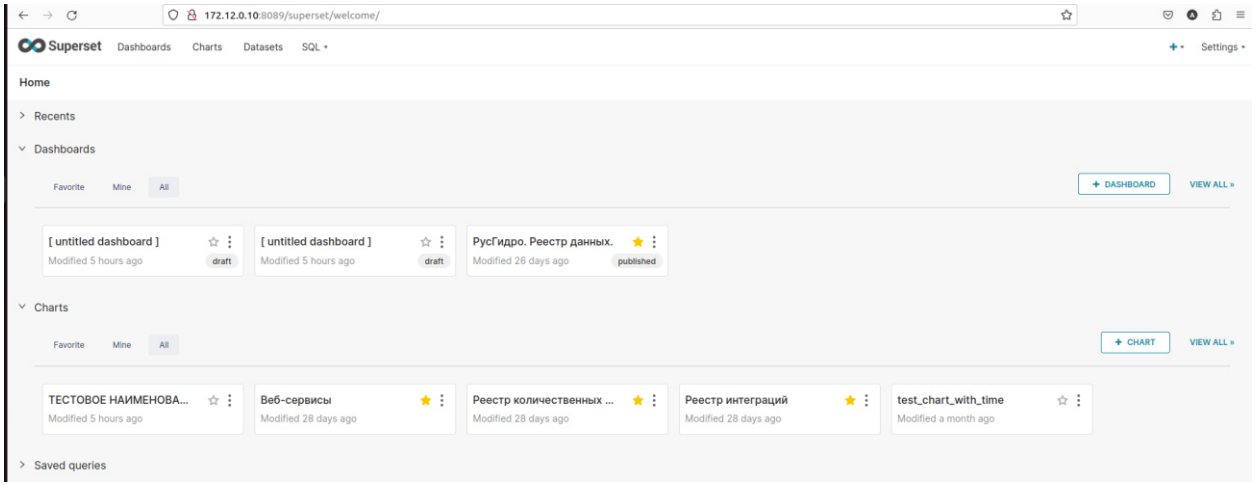
8 Администрирование Apache Superset

8.1 Доступ к компонентам Системы



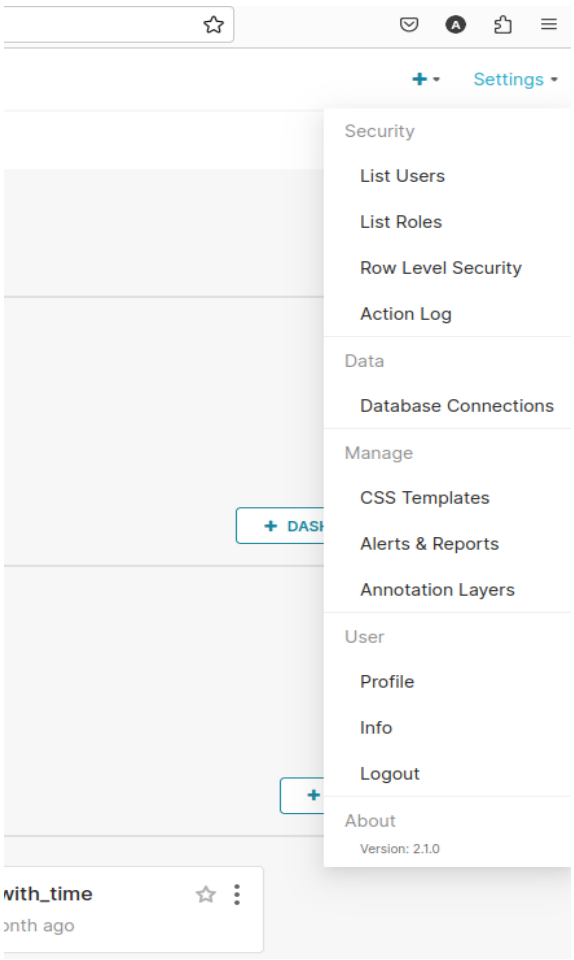
Компоненты Apache Superset можно посмотреть по Web-UI. Для того, чтобы открыть компоненты через Web-UI, необходимо перейти по ссылке http://sr-dc-0057.corp.gidroogk.com:8081:

Далее необходимо ввести логин и пароль (по умолчанию используется admin/admin). Система откроет домашнюю страницу. Так же в Superset реализована система аутентификации LDAP:



Для управления этой компонентой можно выбрать интерактивное меню в правом верхнем углу setting. В нем можно производить:

1. необходимые действия по настройке доступов,
2. управление текущих подключений к внешним СУБД,
3. добавление новых подключений.



Для назначения прав доступа пользователям есть преднастроенные роли.

Подробнее про преднастроенные роли можно изучить в официальной документации по ссылкам:

- https://superset.apache.org/docs/security
- https://github.com/apache/superset/blob/master/RESOURCES/STANDARD_ROLES.md

8.2 Мониторинг работы компонентов Системы

Сначала проверяем доступность web ui. Если она доступна - все работает корректно. Иначе заходим в консоль по ssh доступу на сервер и проверяем работу службы:

```
# docker ps | grep superset

f226a4662177   apachesuperset.docker.scarf.sh/apache/superset:2.1.0
"/app/docker/docker-…"   2 months ago   Up 2 weeks (healthy)   8088/tcp
      superset_worker

0b8f066b2d68   apachesuperset.docker.scarf.sh/apache/superset:2.1.0
"/app/docker/docker-…"   2 months ago   Up 2 weeks              8088/tcp
      superset_worker_beat

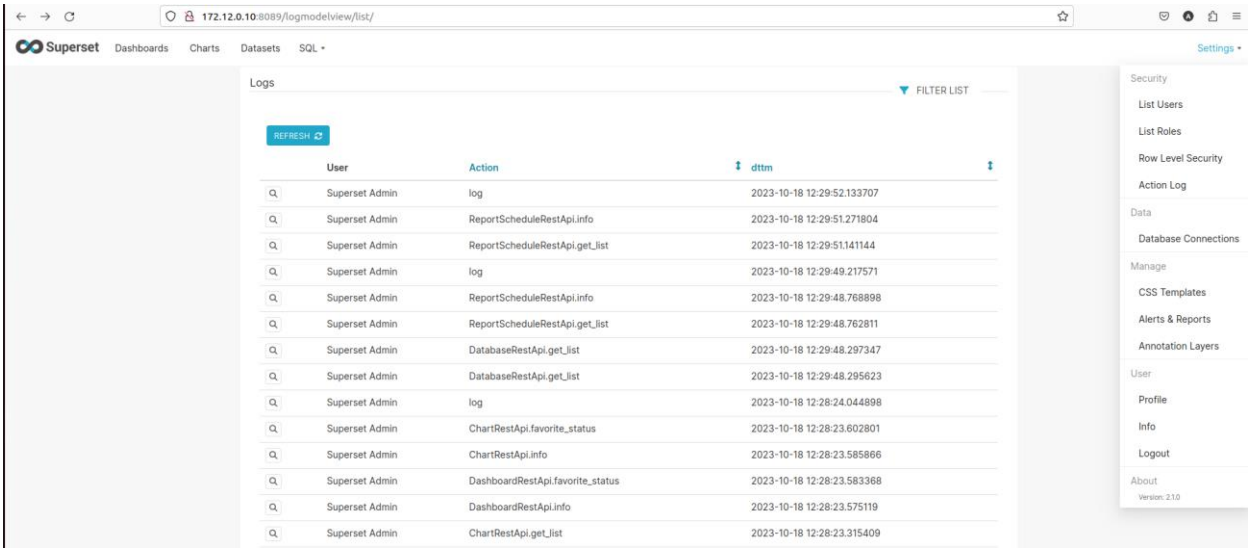
fc0f3b5a95f1   apachesuperset.docker.scarf.sh/apache/superset:2.1.0
"/app/docker/docker-…"   2 months ago   Up 2 weeks (healthy)   0.0.0.0:8089-
>8088/tcp              superset_app

914b97bbec31   redis:7
      "docker-entrypoint.s…"   2 months ago   Up 2 weeks
```

```
      6379/tcp                                superset_cache
```

8.3 Логирование компонентов Системы

Для просмотра текущих логов работы компоненты необходимо перейти в интерактивное меню (правый верхний угол экрана - settings) и выбрать Action Log:



Так как сервис работает в docker контейнере, то логи храняться в контейнере. Управление логами осуществляется через настройку в docker compose, в котором необходимо в конце прописать:

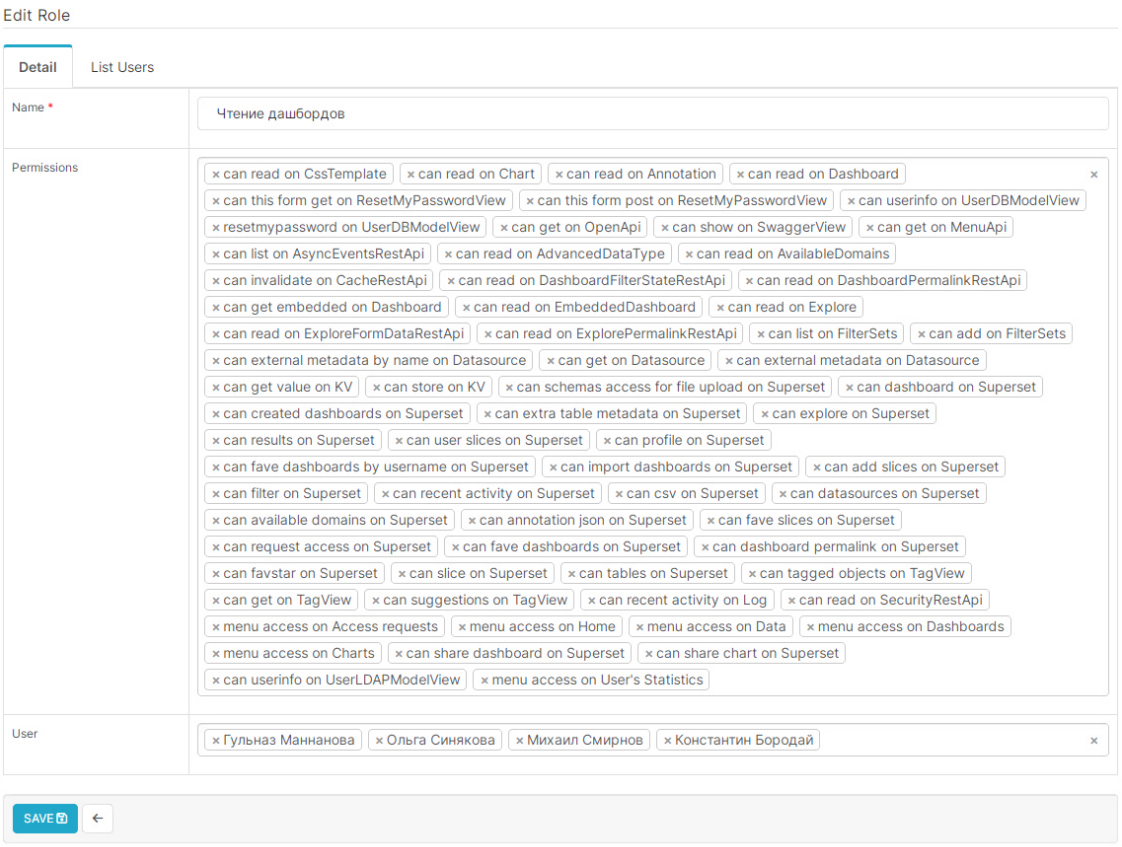
```
logging:
    driver: "json-file"
    options:
       max-file: 5
       max-size: 10m
```

8.4 Настройка ролей

Для настройки и ограничения доступов до каких-либо дашбордов необходимо обладать правами Администратора.

Настройка прав только для просмотра дашбордов производится по следующему алгоритму:

- В Apache SuperSet во вкладке «Settings», «List Roles» настроить роль «Чтение дашбордов» как показано на скрине, выдав необходимые доступы:

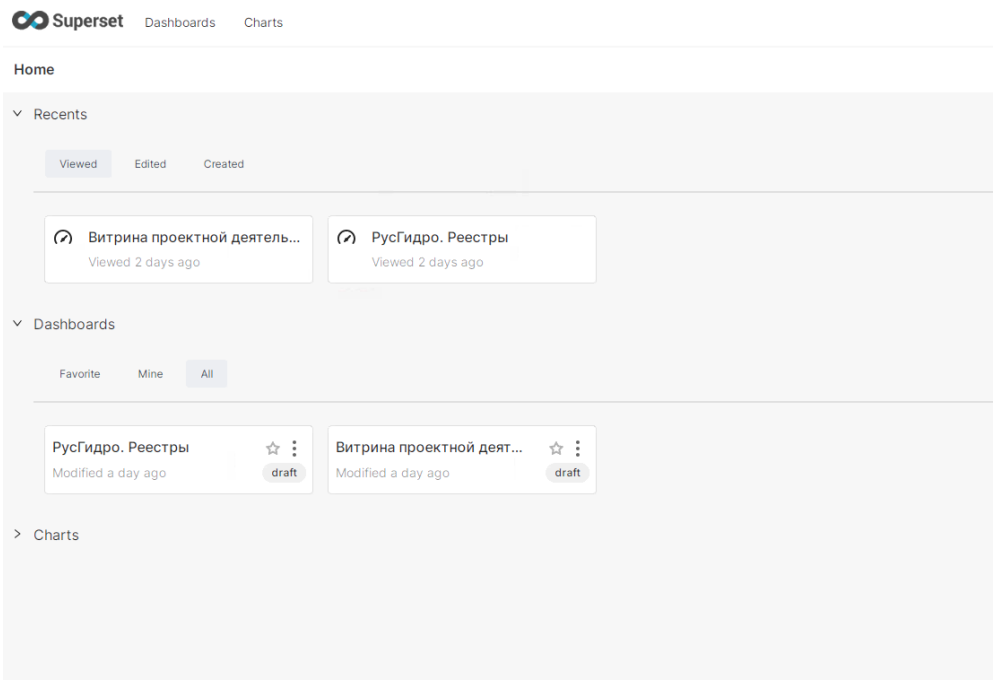


- В Apache SuperSet во вкладке «Settings», «List Roles» настроить роль «Доступ к данным», добавив необходимые источники данных, показанные в дашбордах:



- Во вкладке «Settings», «List Users» необходимо найти пользователя и навесить на него только 2 роли: «Чтение дашбордов» и «Доступ к данным».
- В самом дашборде, который необходимо показать пользователю, в «Owners» добавить пользователя.

После проделанного алгоритма пользователю будет доступно только просмотр данных в определенных дашбордах:



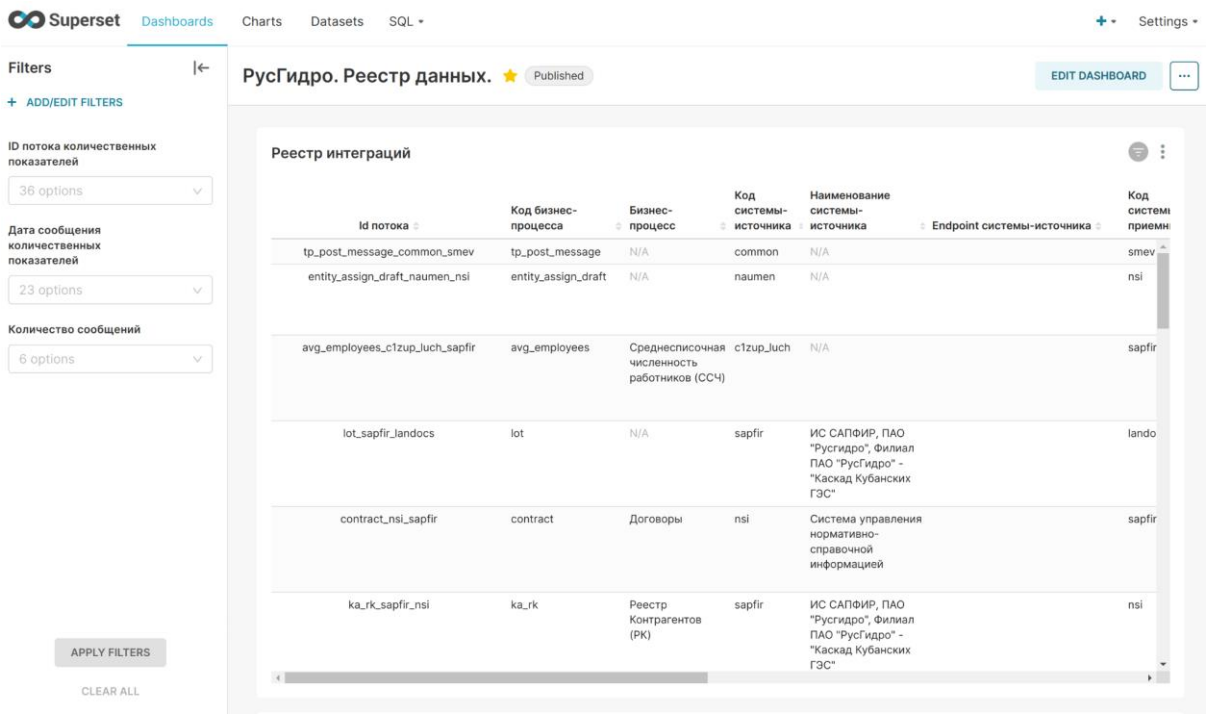
Подробнее о ролях и доступах в Apache SuperSet можно почитать в официальной документации.

8.5 Пользовательские настройки Системы

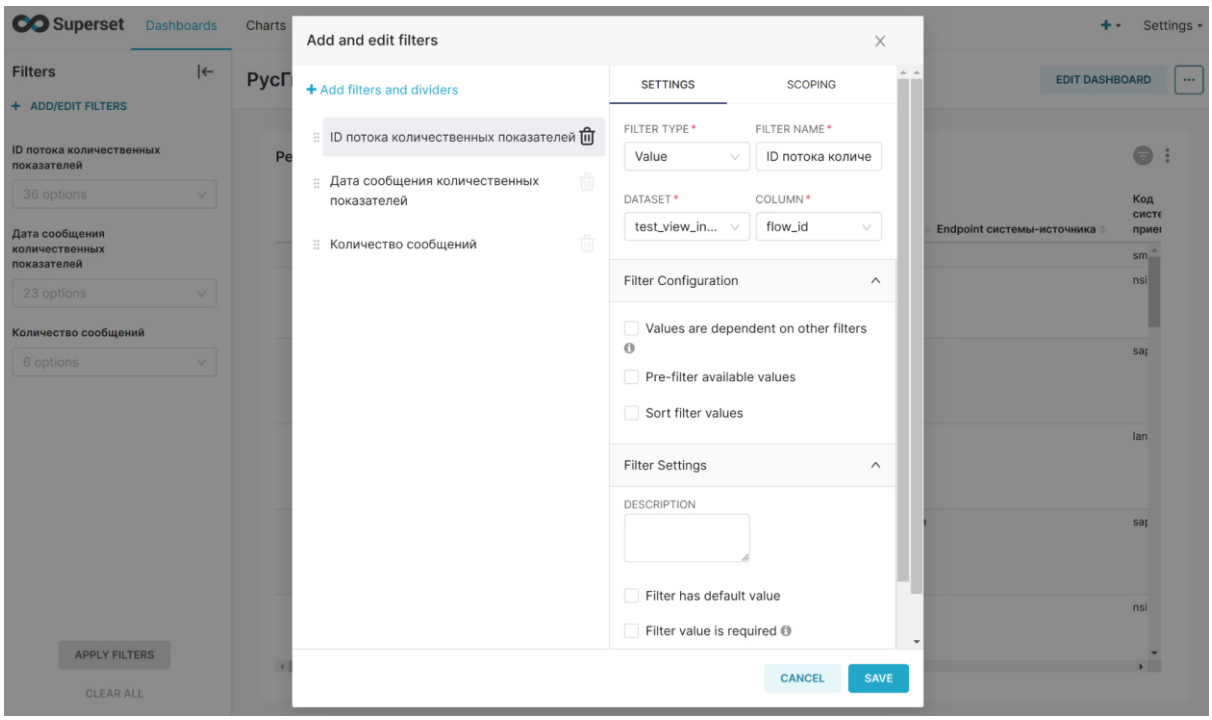
Информация о построении чартов и дашбордов находится в документе “Инструкция по развертыванию Системы” в главе 15.

Настройка фильтров в дашбордах производится по следующему алгоритму:

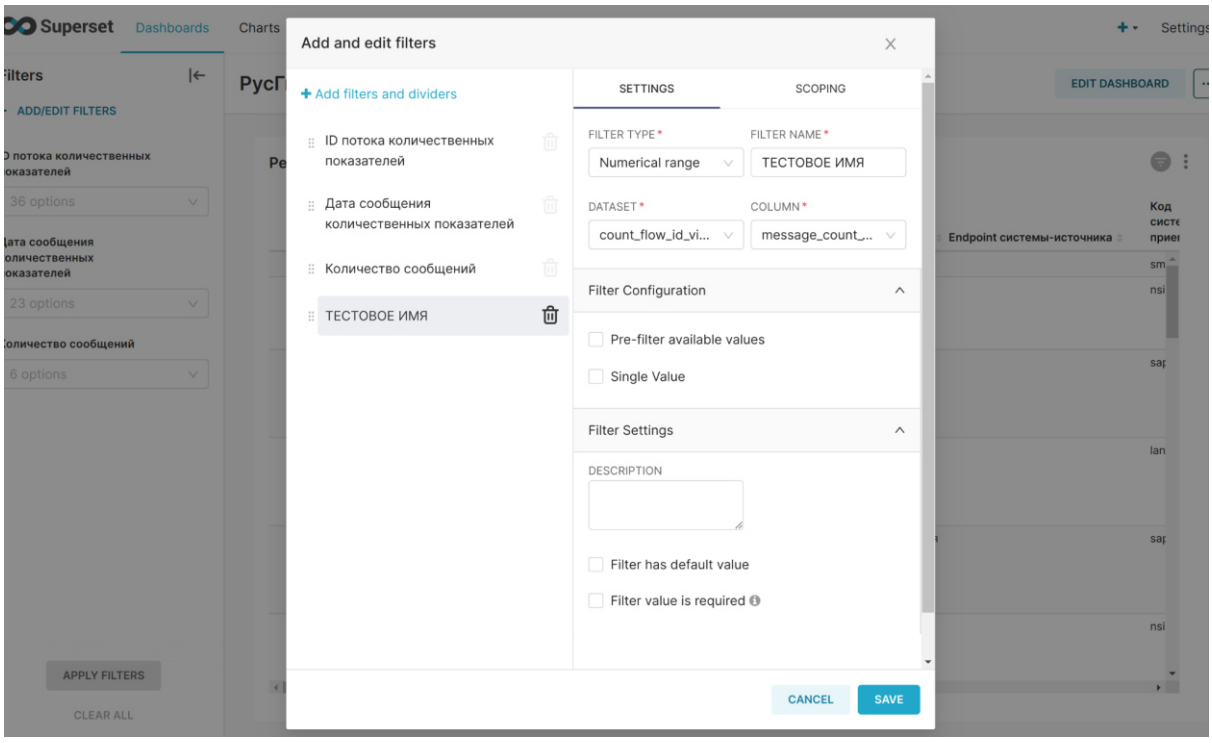
- Необходимо открыть фильтры во вкладке “Dashboards”;



1) Выбрать свойство “Add/Edit Filters”;



- Кликнуть на “Add filter and dividers”;
- Задать необходимые значения;



- Кликнуть на кнопку “Save”.

8.6 Переименование колонок

Для переименования колонок нужно зайти в Чарт, кликнуть на три точки в источник данных, кликнуть на кнопку "Edit dataset" (Рисунок 1). Затем в Columns прописать кастомные названия (Рисунок 2):

Рисунок 1

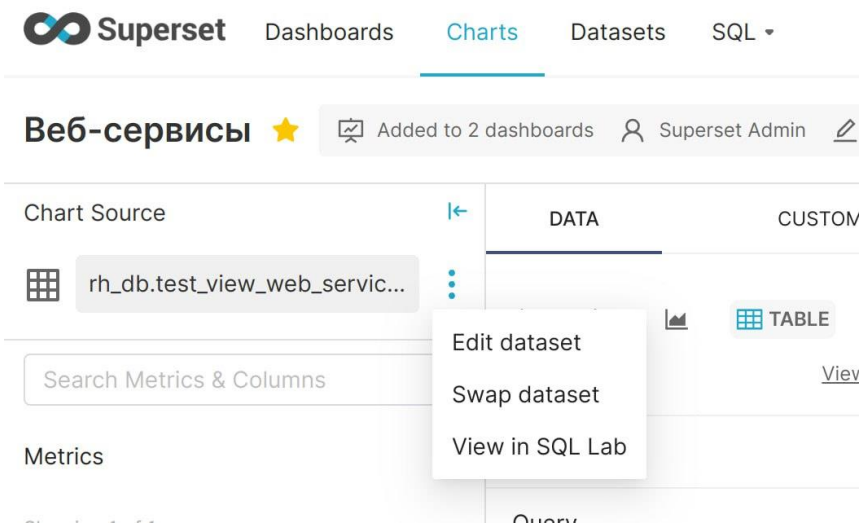
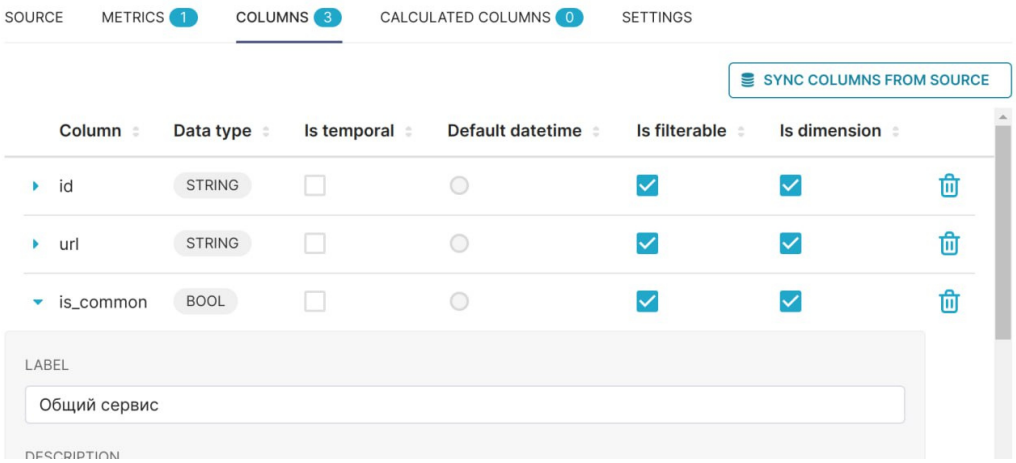


Рисунок 2



Более подробно о работе с Apache Superset можно прочитать в официальной документации - https://superset.apache.org/docs/intro/ .

8.7 Остановка и запуск

Данный сервис работает в докере и для его остановки необходимо выполнить следущие команды:

Остановка:

```
docker compose -f /opt/rh/superset/docker-compose-superset.yml down
```

Запуск:

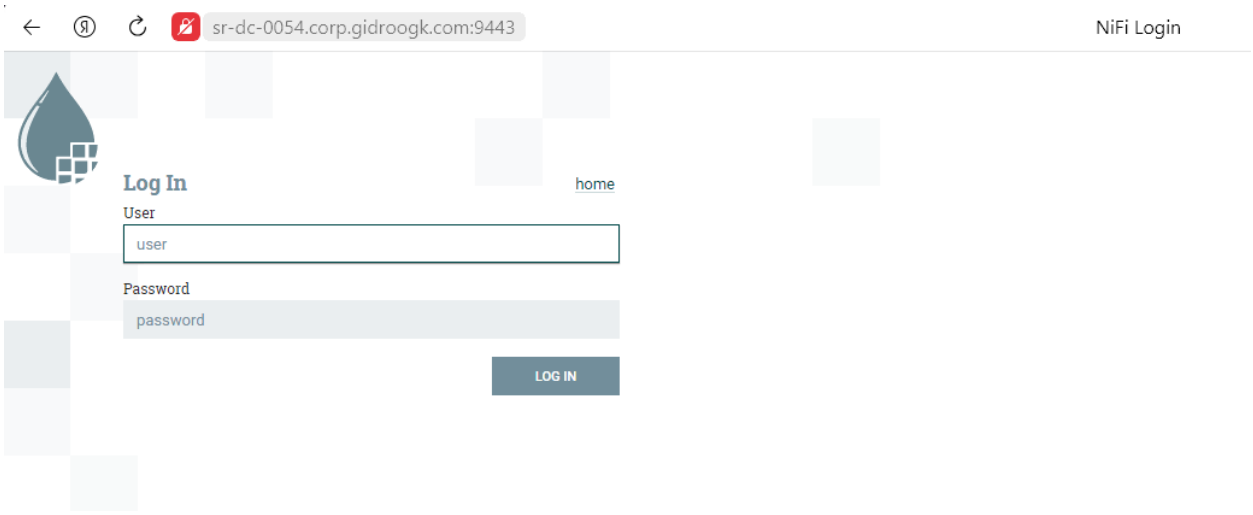
```
docker compose -f /opt/rh/superset/docker-compose-superset.yml down up -d
```

9 Администрирование Apache NiFi

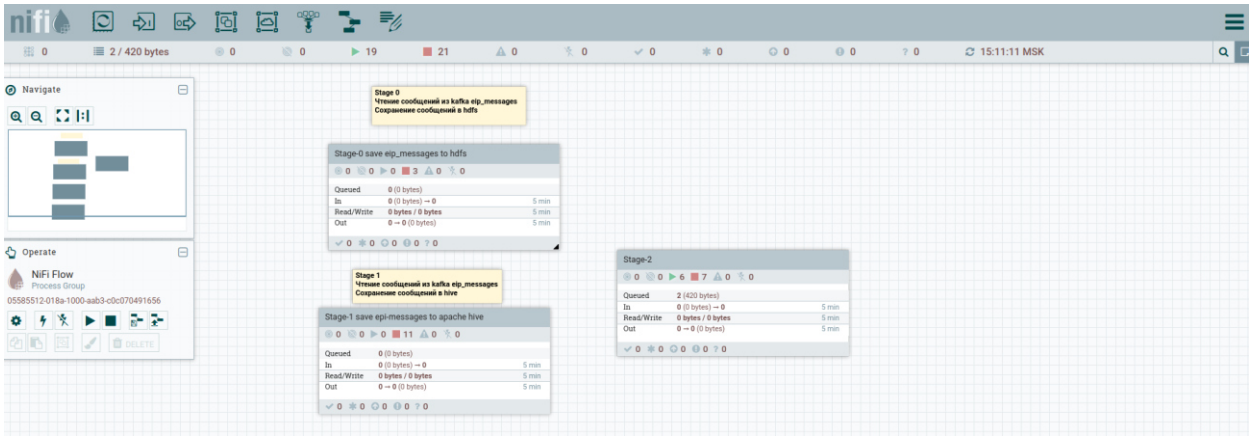
9.1 Доступ к компонентам Системы

Компоненты Hadoop NiFi можно посмотреть по Web-UI. Для того чтобы открыть компоненты через Web-UI необходимо перейти по ссылке http://sr-dc-0054.corp.gidroogk.com:8443.

После открытия данного адреса необходимо будет пройти авторизацию:



Учетные записи берутся из LDAP. После успешной аутентификации вы увидите в браузере окно со всей информацией о текущем состоянии кластера, пример:



9.2 Мониторинг работы компонентов Системы

Мониторинг работы основных систем компоненты можно наблюдать через основное меню.

Для проверки работоспособности сервисов необходимо перейти на сервер по ssh и вызвать команду для прроверки состояния работы сервиса

```
$ systemctl status nifi.service
```

Для перезапуска сервиса используеутся команда

```
$ systemctl restart nifi.service
```

Остановка сервиса производится командой

```
$ systemctl stop nifi.service
```

Запуск сервиса производиться командой

```
$ systemctl start nifi.service
```

9.3 Логирование компонентов Системы

Основные логи работы компоненты можно просмотреть через консоль сервера с компонентой. Все логи находятся по пути установки самой компоненты:

```
# ls -la /usr/local/apache/nifi/logs/
```

Основные логи имеют содержат названия:

- nifi-app.log
- nifi-bootstrap.log
- nifi-user.log

Ротация логов производится самим сервисом. Для настройки ротации необхходимо отредактировать файл конфигурации /usr/local/apache/nifi/logback.xml:

```
    <appender name="APP_FILE"
class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>${org.apache.nifi.bootstrap.config.log.dir}/nifi-app.log</file>
        <rollingPolicy
class="ch.qos.logback.core.rolling.SizeAndTimeBasedRollingPolicy">
            <!--
              For daily rollover, use 'app_%d.log'.
              For hourly rollover, use 'app_%d{yyyy-MM-dd_HH}.log'.
              To GZIP rolled files, replace '.log' with '.log.gz'.
              To ZIP rolled files, replace '.log' with '.log.zip'.
            -->

<fileNamePattern>${org.apache.nifi.bootstrap.config.log.dir}/nifi-
app_%d{yyyy-MM-dd_HH}.%i.log</fileNamePattern>
            logback.xml
            <!-- keep 30 log files worth of history -->
            <maxHistory>30</maxHistory>
        </rollingPolicy>
        <immediateFlush>true</immediateFlush>
        <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
            <pattern>%date %level [%thread] %logger{40} %msg%n</pattern>
        </encoder>
    </appender>


    <appender name="USER_FILE"
class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>${org.apache.nifi.bootstrap.config.log.dir}/nifi-
```

```
user.log</file>
        <rollingPolicy
class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <!--
              For daily rollover, use 'user_%d.log'.
              For hourly rollover, use 'user_%d{yyyy-MM-dd_HH}.log'.
              To GZIP rolled files, replace '.log' with '.log.gz'.
              To ZIP rolled files, replace '.log' with '.log.zip'.
            -->

<fileNamePattern>${org.apache.nifi.bootstrap.config.log.dir}/nifi-
user_%d.log</fileNamePattern>
            <!-- keep 30 log files worth of history -->
            <maxHistory>30</maxHistory>
        </rollingPolicy>
        <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
            <pattern>%date %level [%thread] %logger{40} %msg%n</pattern>
        </encoder>
    </appender>


    <appender name="BOOTSTRAP_FILE"
class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>${org.apache.nifi.bootstrap.config.log.dir}/nifi-
bootstrap.log</file>
        <rollingPolicy
class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
            <!--
              For daily rollover, use 'bootstrap_%d.log'.
              For hourly rollover, use 'bootstrap_%d{yyyy-MM-dd_HH}.log'.
              To GZIP rolled files, replace '.log' with '.log.gz'.
              To ZIP rolled files, replace '.log' with '.log.zip'.
            -->

<fileNamePattern>${org.apache.nifi.bootstrap.config.log.dir}/nifi-
bootstrap_%d.log</fileNamePattern>
            <!-- keep 5 log files worth of history -->
            <maxHistory>5</maxHistory>
        </rollingPolicy>
        <encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">
            <pattern>%date %level [%thread] %logger{40} %msg%n</pattern>
        </encoder>
    </appender>
```

Во всех этих разделах есть пункты:

- <maxFileSize>100MB</maxFileSize> - указывается размер хранимых логов;
- <maxHistory>30</maxHistory> - указывает как долго (сколько дней) хранить логи, по достижению одного из максимумов будет перезатирание более старых логов. Так же раз в сутки идет отрезание текущего лога и переименование файлы в формат;
- <название лога>-<дата лога>.log, например, nifi-app_2023-11-30_14.0.log.

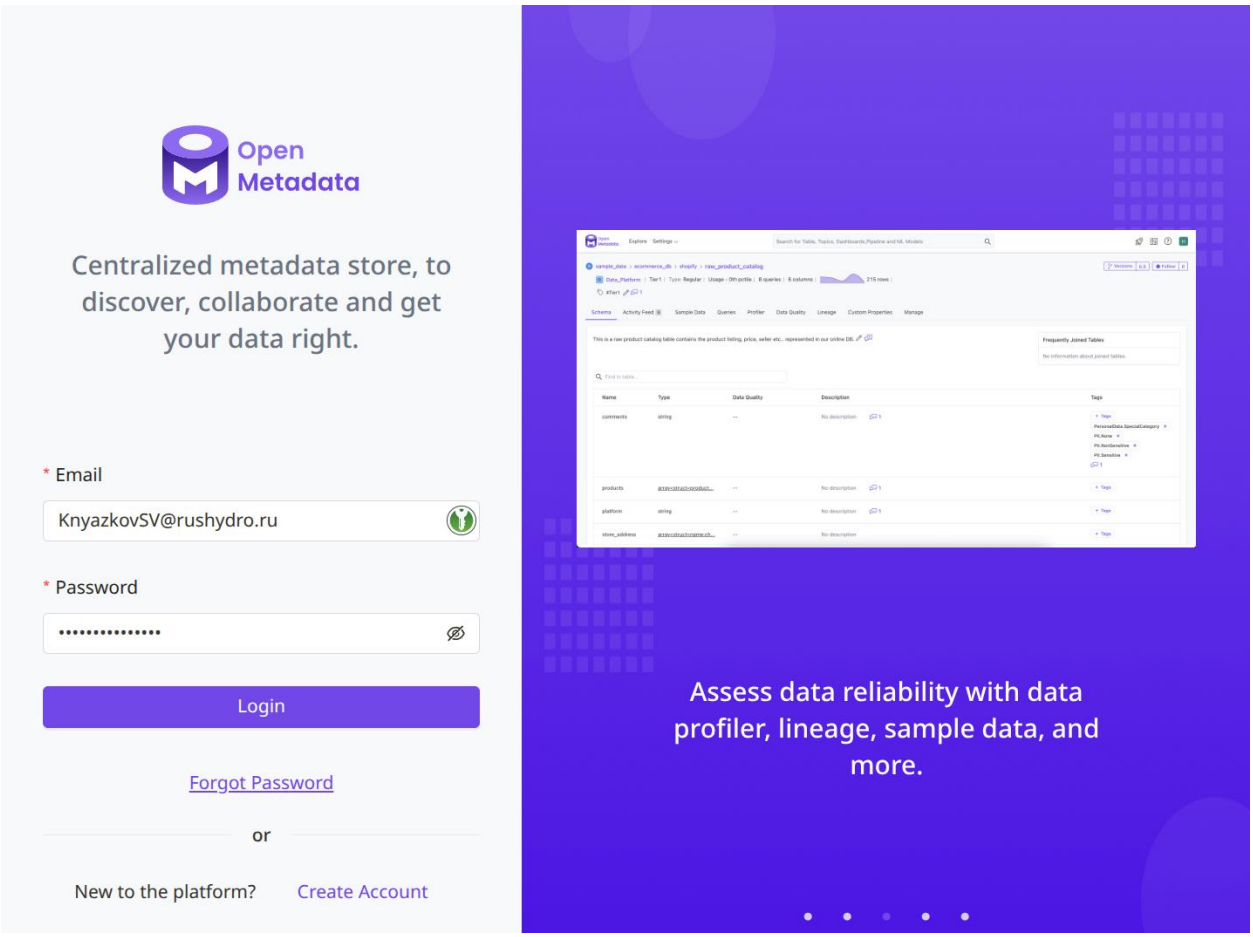
10 Администрирование OpenMetadata

10.1 Доступ к компонентам Системы

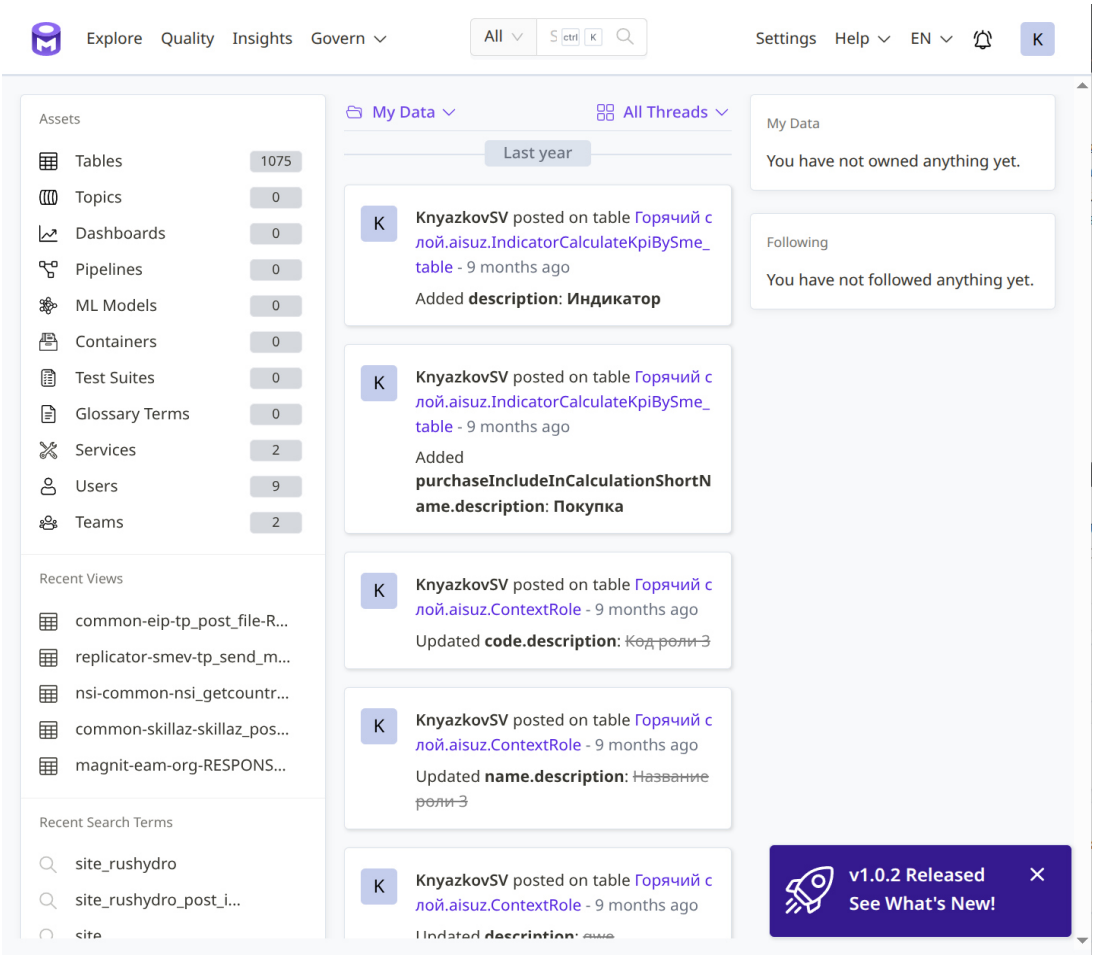
Компонента OpenMetdata состоят из нескольких частей:

- OpenMetadata ingestion
- OpenMetadata server
- OpenMetadata elasticsearch

Для доступа по Web-UI к OpenMetadata ingestion необходимо перейти по ссылке http://sr-dc-0058:8585. После открытия данного адресы вы увидите форму для ввода логина и пароля:



В форме вводим email для учетной записи LDAP и доменный пароль. Пройдя авторизацию - для вас откроется окно управления сервисом компоненты:

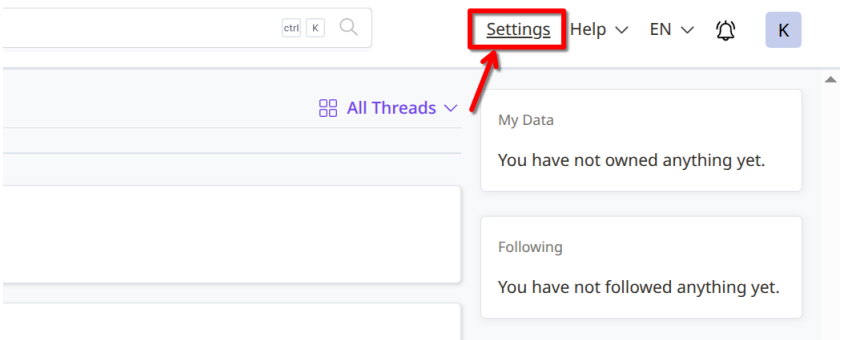


10.2 Назначение ролей

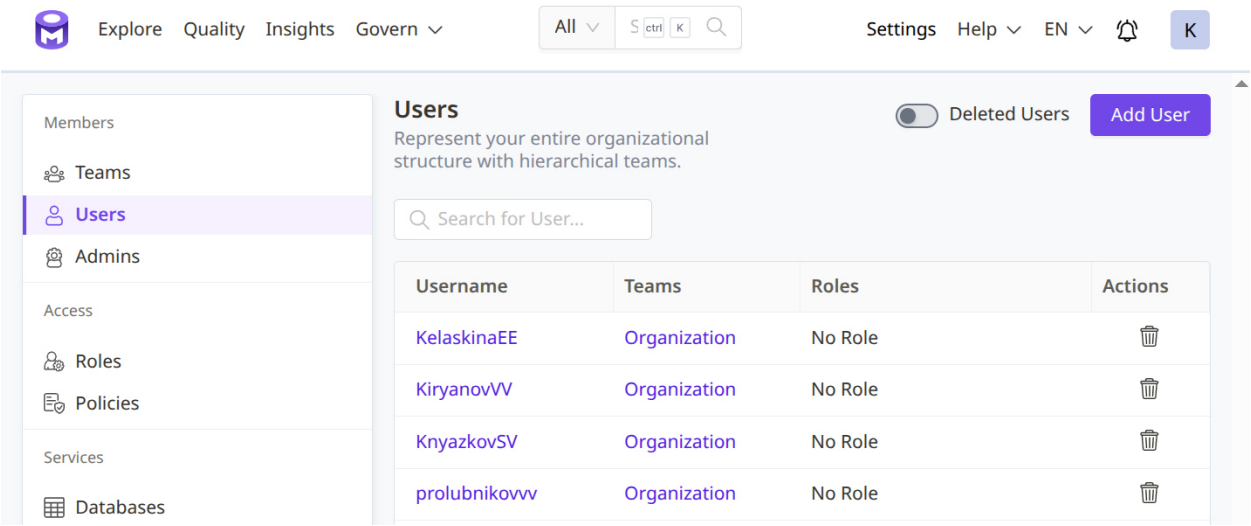
Для настройки и ограничения доступов до каких-либо дашбордов необходимо обладать правами Администратора. По умолчанию в OpenMetadata настроены роли, которые можно брать за основу.

Настройка прав пользователям для изменения описания атрибутов настраивается следующим образом:

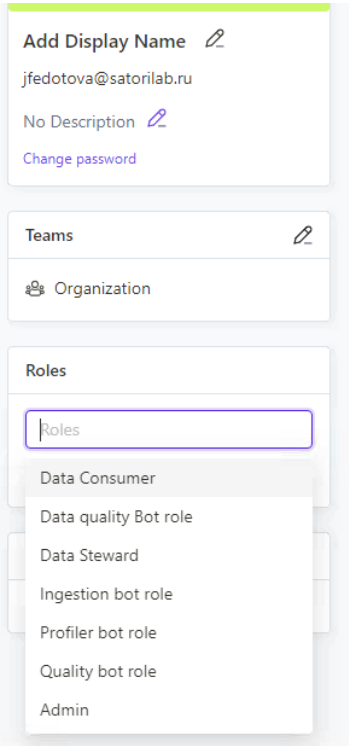
- необходимо кликнуть на “Settings” в левом верхнем углу;

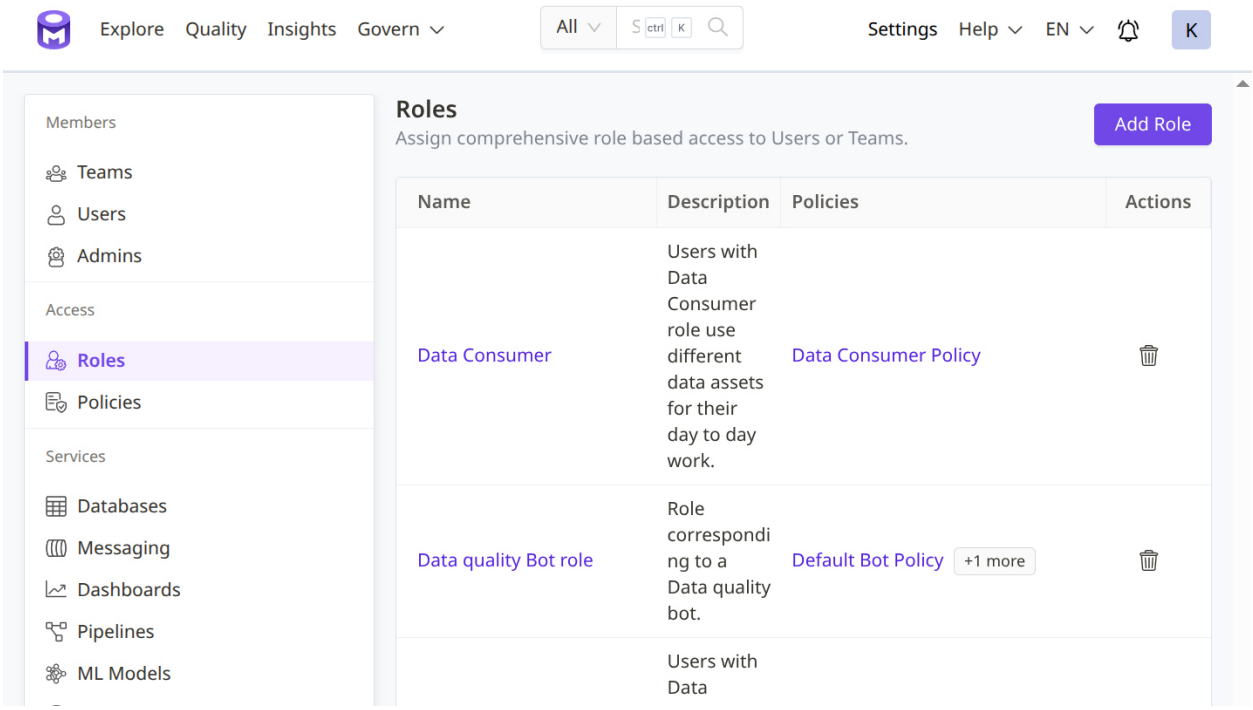


- выбрать вкладку “Users”;



- найти пользователя и кликнуть на него;
- добавить роль “Data Consumer”.



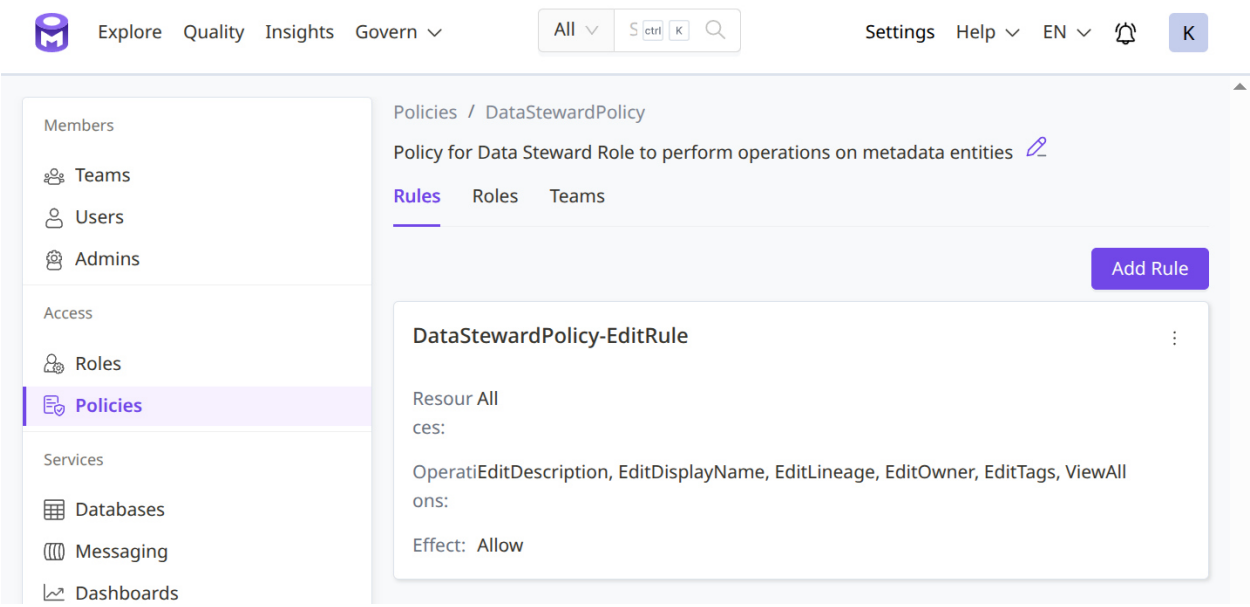


Настройка списка ролей выполняется в разделе Roles:

Для создания собственной роли необходимо в “Settings” настроить “Policy” и “Role”, добавив необходимые разрешения. Например, ниже приведен пример политики, позволяющей включить ее для Data Stewards с использованием двух правил:

- разрешить операции с глоссарием: включает политику, разрешающую операции со всеми действиями, связанными с глоссарием;
- правило редактирования: предоставляет доступ к Data Steward для редактирования описания и тегов всех объектов; предоставление пользователю возможности управлять данными.

–



Вы можете настроить эти разрешения в соответствии с потребностями вашей организации. Более подробно о ролях и доступах можно почитать в официальной документации:

- https://docs.open-metadata.org/v1.1.x/how-to-guides/admin-guide-roles-policies/use-cases.

10.3 Мониторинг работы компонентов Системы

Самое быстрое это проверяем доступность web ui. Если она доступна - все ок. Иначе заходим в консоль по ssh доступу на сервер и проверяем работу службы:

```
# docker ps | grep openmetadata

febfdb28ace4   docker.getcollate.io/openmetadata/ingestion:1.0.4
      "/bin/bash /opt/airf…"   5 weeks ago      Up 2 weeks
      0.0.0.0:8090->8080/tcp              openmetadata_ingestion

dc7aacbcd9b2   docker.getcollate.io/openmetadata/server:1.0.4
      "./openmetadata-star…"   5 weeks ago      Up 2 weeks (healthy)
0.0.0.0:8585-8586->8585-8586/tcp   openmetadata_server

d81fe03355c5   docker.elastic.co/elasticsearch/elasticsearch:7.10.2   "/tini
-- /usr/local…"   5 weeks ago Up 2 weeks              9200/tcp, 9300/tcp
      openmetadata_elasticsearch
```

так же состояние можно проверить командой:

```
docker compose -f /opt/rh/openmetadata/docker-compose-openmetadata.yml ps
```

со следующим выводом:

```
[root@sr-dc-0058 niyazova@corp.gidroogk.com]# docker compose -f /opt/rh/openmetadata/docker-compose-openmetadata.yml ps
NAME                        COMMAND                  SERVICE             STATUS              PORTS
openmetadata_elasticsearch   "/tini -- /usr/local…"   elasticsearch       running             9200/tcp, 9300/tcp
openmetadata_ingestion       "/bin/bash /opt/airf…"   ingestion           running             0.0.0.0:8090->8080/tcp
openmetadata_server          "./openmetadata-star…"   openmetadata-server   running (healthy)   0.0.0.0:8585-8586->8585-8586/tcp
[root@sr-dc-0058 niyazova@corp.gidroogk.com]#
```

Status Running означает что сервис функционирует нормально. В других случая необходимо смотерть логи.

Для остановки сервиса необходимо вызвать команду:

```
docker compose -f /opt/rh/openmetadata/docker-compose-openmetadata.yml down
```

Для запуска:

```
docker compose -f /opt/rh/openmetadata/docker-compose-openmetadata.yml up -d
```

10.4 Логирование компонентов Системы

Основные логи работы компоненты можно посмотреть в консоли сервера установленной компоненты командами:

```
# docker logs openmetadata_server -n 100
# docker logs openmetadata_ingestion -n 100
# docker logs openmetadata_elasticsearch -n 100
```

где -n 100 выводит последние 100 записей из лога в контейнере docker. Так как сервис работает в docker-контейнере, то логи храняться в контейнере. Управление логами осуществляется через корректировку файла docker-compose-openmetadata.yml, необходимо в конце прописать:

```
logging:
    driver: "json-file"
    options:
       max-file: 5
       max-size: 10m
```

Для самих сервисов параметры логирования прописаны в конфигурационном файле внутри контейнера и имеют следующие значения:

```
  appenders:
    - type: console
      threshold: TRACE
      logFormat: "%level [%d{ISO8601,UTC}] [%t] %logger{5} - %msg%n"
      timeZone: UTC
    - type: file
      filterFactories:
        - type: audit-exclude-filter-factory
      threshold: TRACE
      logFormat: "%level [%d{ISO8601,UTC}] [%t] %logger{5} - %msg%n"
      currentLogFilename: ./logs/openmetadata.log
      archivedLogFilenamePattern: ./logs/openmetadata-%d{yyyy-MM-dd}-
%i.log.gz
      archivedFileCount: 7
      timeZone: UTC
      maxFileSize: 50MB
    - type: file
      filterFactories:
        - type: audit-only-filter-factory
      threshold: TRACE
      logFormat: "%level [%d{ISO8601,UTC}] [%t] %logger{5} - %msg%n"
```

```
    currentLogFilename: ./logs/audit.log
    archivedLogFilenamePattern: ./logs/audit-%d{yyyy-MM-dd}-%i.log.gz
    archivedFileCount: 25
    timeZone: UTC
    maxFileSize: 50MB
```

11 Администрирование ClickHouse

11.1 Доступ к компонентам Системы

Основной доступ и работа с компонентой выполняется в консоли. Подключаемся по ssh к серверам sr-dc-0052.corp.gidroogk.com и sr-dc-00143.corp.gidroogk.com с установленной компонентой и запускаем консольного клиента по работе с СУБД clickhouse-client:

```
# clickhouse-client  -u default -h 127.0.0.1
ClickHouse client version 23.3.3.52 (official build).
Connecting to 127.0.0.1:9000 as user default.
```

далее вводим пароль и работаем уже в клиенте:

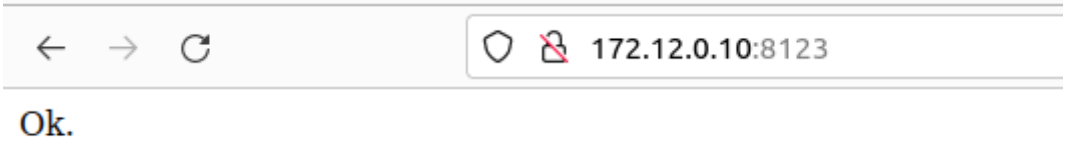
```
Password for user (default):
Connecting to 127.0.0.1:9000 as user default.
Connected to ClickHouse server version 23.3.3 revision 54462.
Warnings:
 * Linux transparent hugepages are set to "always". Check
/sys/kernel/mm/transparent_hugepage/enabled
```

С методами создания пользователей, ролей и назначение ролей можно ознокмится в официальной документации по ссылкам:

- https://clickhouse.com/docs/ru/operations/access-rights
- https://presentations.clickhouse.com/meetup44/RBAC.pdf

11.2 Мониторинг работы компонентов Системы

Проверить состояние работы кластера компоненты можно обратившись к сервисному порту 8123:



Такой вывод означает стабильную работу компоненты.

Так же можно проверить работу службы:

```
# systemctl status clickhouse-server.service
clickhouse-server.service - ClickHouse Server (analytic DBMS for big data)
   Loaded: loaded (/usr/lib/systemd/system/clickhouse-server.service;
enabled; vendor preset: disabled)
   Active: active (running) since Ср 2023-10-25 10:51:11 MSK; 5 days ago
 Main PID: 16257 (clickhouse-serv)
      Tasks: 371
   Memory: 711.4M
   CGroup: /system.slice/clickhouse-server.service
```

```
           ├─16237 clickhouse-watchdog --config=/etc/clickhouse-
server/config.xml --pid-file=/run/clickhouse-server/clickhouse-server.pid
           └─16257 /usr/bin/clickhouse-server --config=/etc/clickhouse-
server/config.xml --pid-file=/run/clickhouse-server/clickhouse-server.pid


окт 25 10:51:10 rg00infra01 clickhouse-server[16237]: Saved preprocessed
configuration to
'/opt/rh/yandex/clickhouse/data/clickhouse/preprocessed_configs/config.xml'.
окт 25 10:51:10 rg00infra01 clickhouse-server[16237]: Processing
configuration file '/etc/clickhouse-server/users.xml'.
окт 25 10:51:10 rg00infra01 clickhouse-server[16237]: Merging configuration
file '/etc/clickhouse-server/users.d/default-password.xml'.
окт 25 10:51:10 rg00infra01 clickhouse-server[16237]: Saved preprocessed
configuration to
'/opt/rh/yandex/clickhouse/data/clickhouse/preprocessed_configs/users.xml'.
окт 25 10:51:11 rg00infra01 systemd[1]: Started ClickHouse Server (analytic
DBMS for big data).
окт 27 13:32:48 rg00infra01 clickhouse-server[16237]: Processing
configuration file '/etc/clickhouse-server/users.xml'.
окт 27 13:32:48 rg00infra01 clickhouse-server[16237]: Saved preprocessed
configuration to
'/opt/rh/yandex/clickhouse/data/clickhouse/preprocessed_configs/users.xml'.
окт 27 13:33:54 rg00infra01 clickhouse-server[16237]: Processing
configuration file '/etc/clickhouse-server/users.xml'.
окт 27 13:33:54 rg00infra01 clickhouse-server[16237]: Merging configuration
file '/etc/clickhouse-server/users.d/default-password.xml'.
окт 27 13:33:54 rg00infra01 clickhouse-server[16237]: Saved preprocessed
configuration to
'/opt/rh/yandex/clickhouse/data/clickhouse/preprocessed_configs/users.xml'.
```

Остановка кластера происходит с помощью команды остановки на каждой ноде:

```
systemctl stop clickhouse-server.service
```

Запуск осуществляется командой:

```
systemctl start clickhouse-server.service
```

Для проверки работоспособности кластера в части синхронизации можно выполнить тестую запись таблицы на одной ноде и прочитать ее на второй.

11.3 Логирование компонентов Системы

Все основные логи работы компоненты можно найти в директории указанной в настройках компоненты в файле /etc/clickhouse-server/config.xml в разделе:

```
<clickhouse>
    <logger>
        <log>/var/log/clickhouse-server/clickhouse-server.log</log>
```

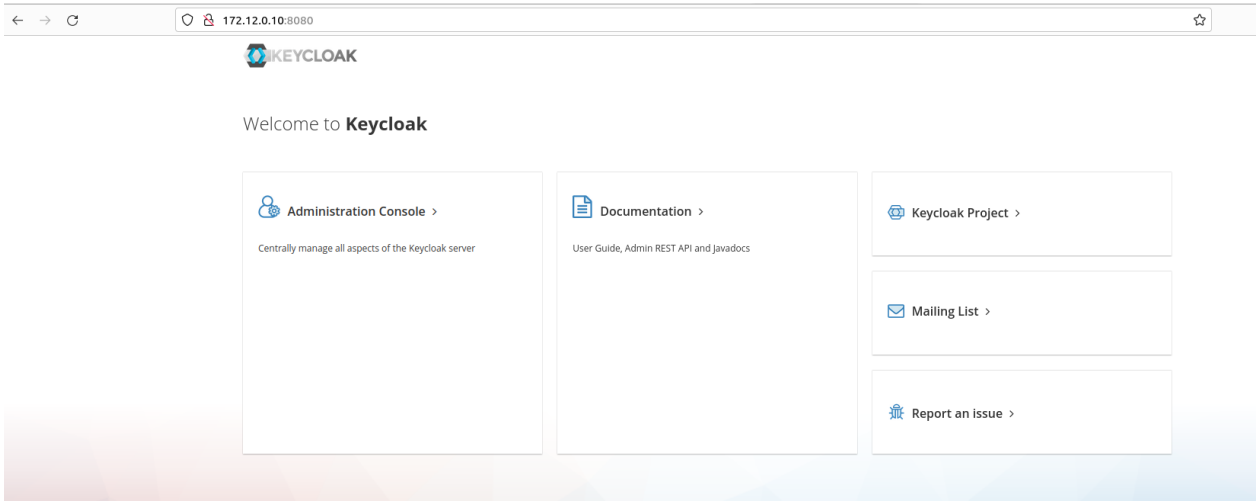
Так же в данном разделе указаны параметры ротации логов:

```
<size>1000M</size>
<count>10</count>
```

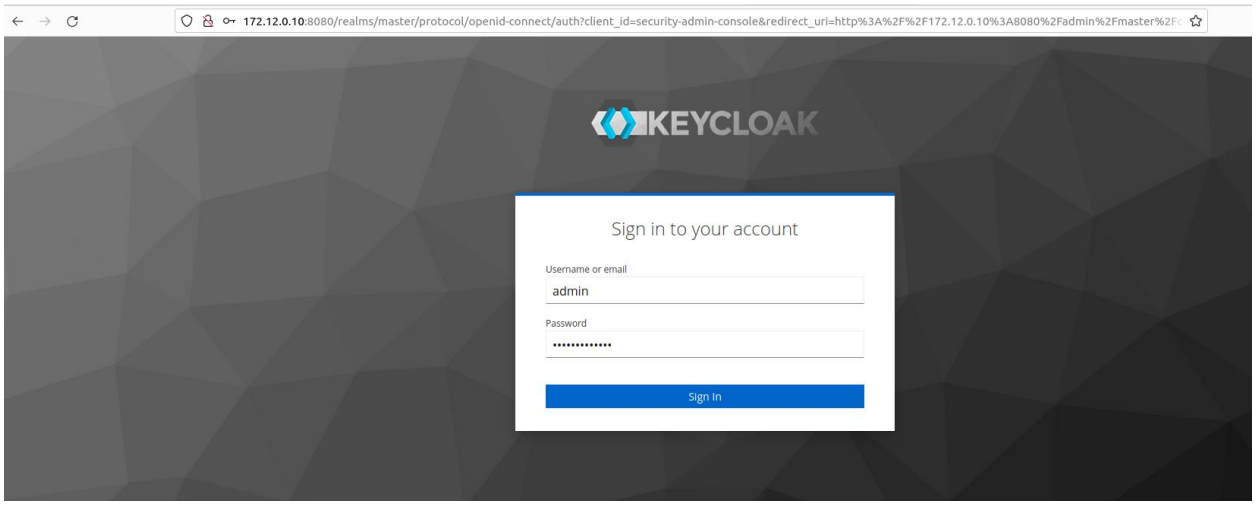
12 Администрирование KeyCloak

12.1 Доступ к компонентам Системы

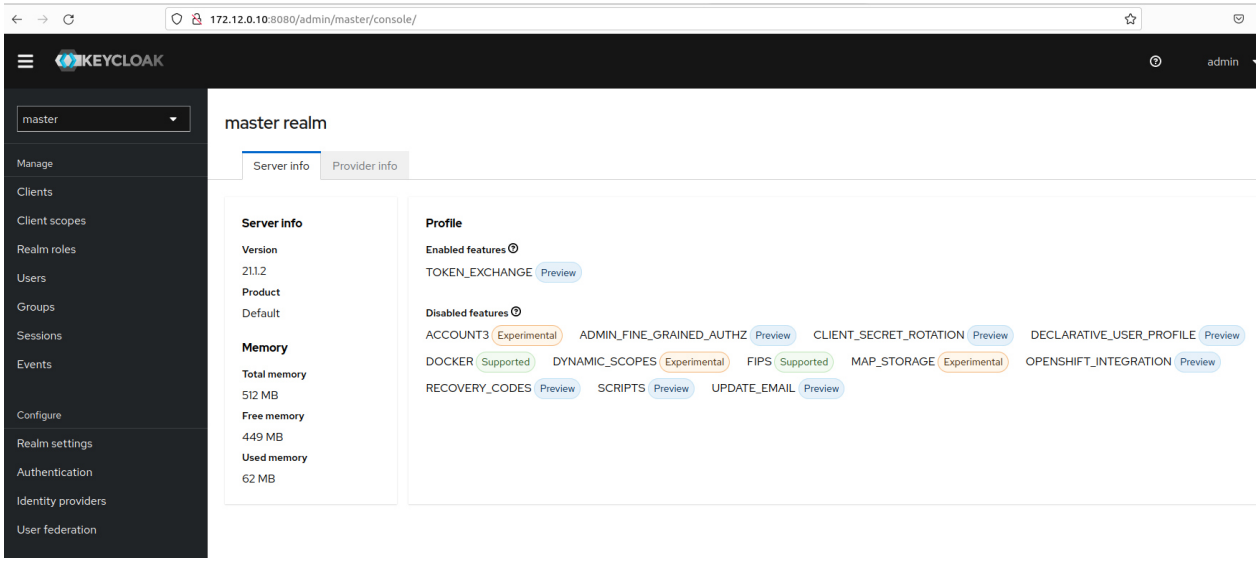
Компоненты KeyCloak можно посмотреть по Web-UI. Для того чтобы открыть компоненты через Web-UI необходимо перейти по ссылке http://localhost:8080 - здесь необходимо заменить localhost на ip адрес вашего сервера с установленной компонентой. После открытия данного адресы вы увидите в браузере окно:



далее выбираем Administration Console и после перехода на страницу аутентификации вводим логин/пароль (по умолчанию admin/keycloakadmin):



после успешной аутентификации видим панель администратора:



В данной панели мы можем полностью управлять компонентой.

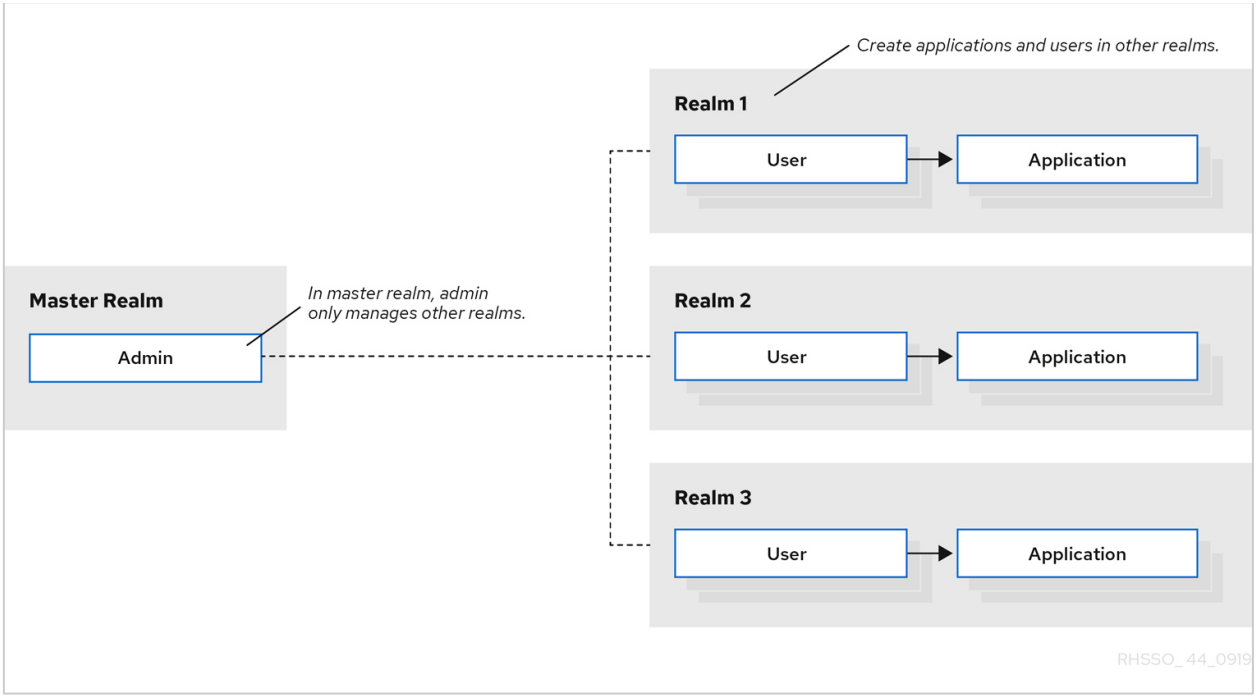
12.2 Управление областями

12.2.1 Master realm

В консоли администратора существует два типа областей:

- Master realm: Эта область была создана для вас, когда вы впервые запустили Keycloak. Она содержит учетную запись администратора, которую вы создали при первом входе в систему. Используйте главную область только для создания областей в вашей системе и управления ими.
- Other realms: Эти области создаются администратором в главной области. В этих сферах администраторы управляют пользователями необходимыми для приложений. Приложения принадлежат пользователям.

Области и приложения:



Области изолированы друг от друга и могут управлять и аутентифицировать только тех пользователей, которых они контролируют. Следование этой модели безопасности помогает предотвратить случайные изменения и следует традиции предоставления учетным записям пользователей доступа только к тем привилегиям и полномочиям, которые необходимы для успешного выполнения их текущей задачи.

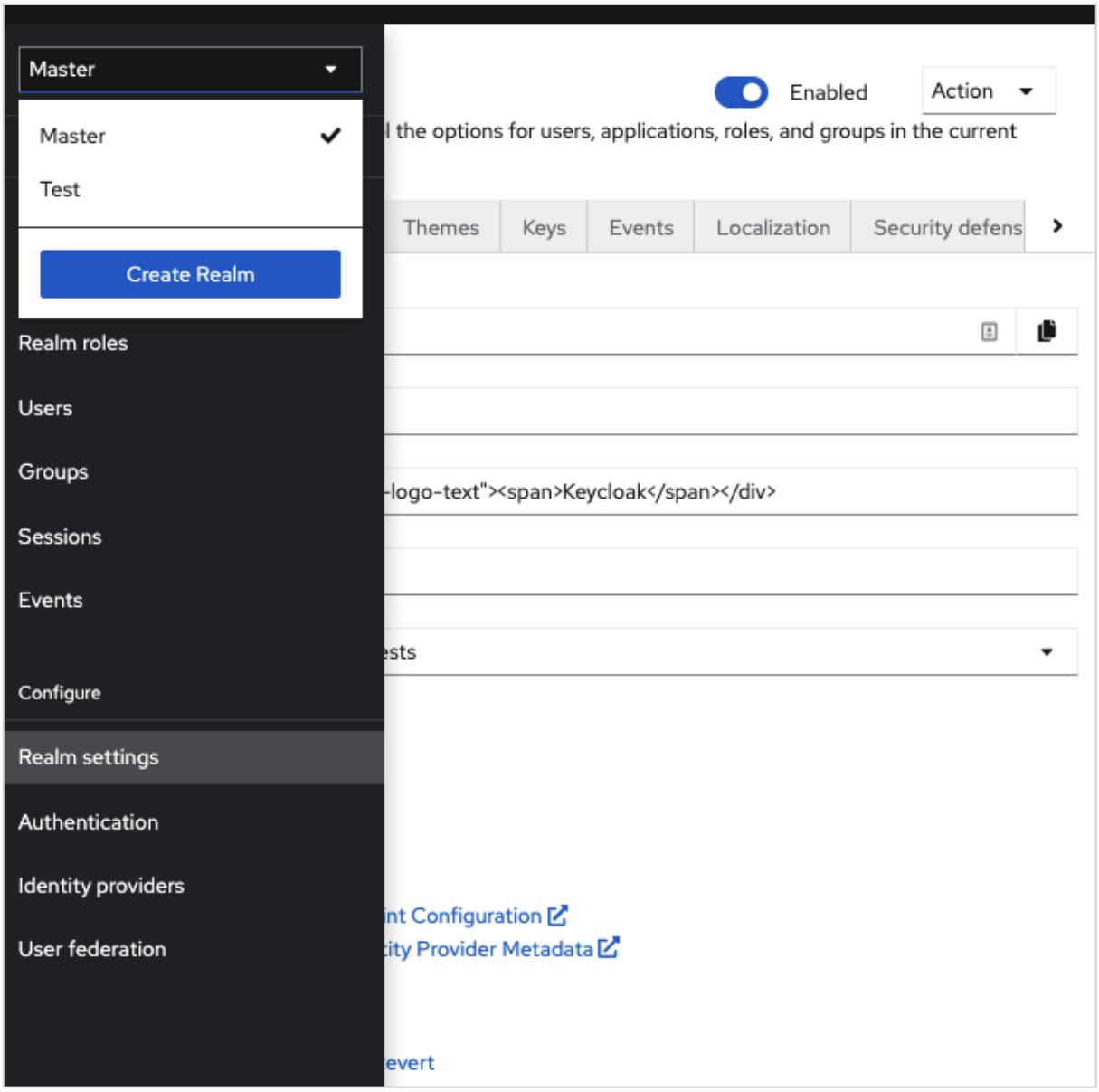
12.2.2 Создание области

Вы создаете область, чтобы предоставить пространство управления, где вы можете создавать пользователей и давать им разрешения на использование приложений. При первом входе в систему вы обычно находитесь в главной области, области верхнего уровня, из которой вы создаете другие области.

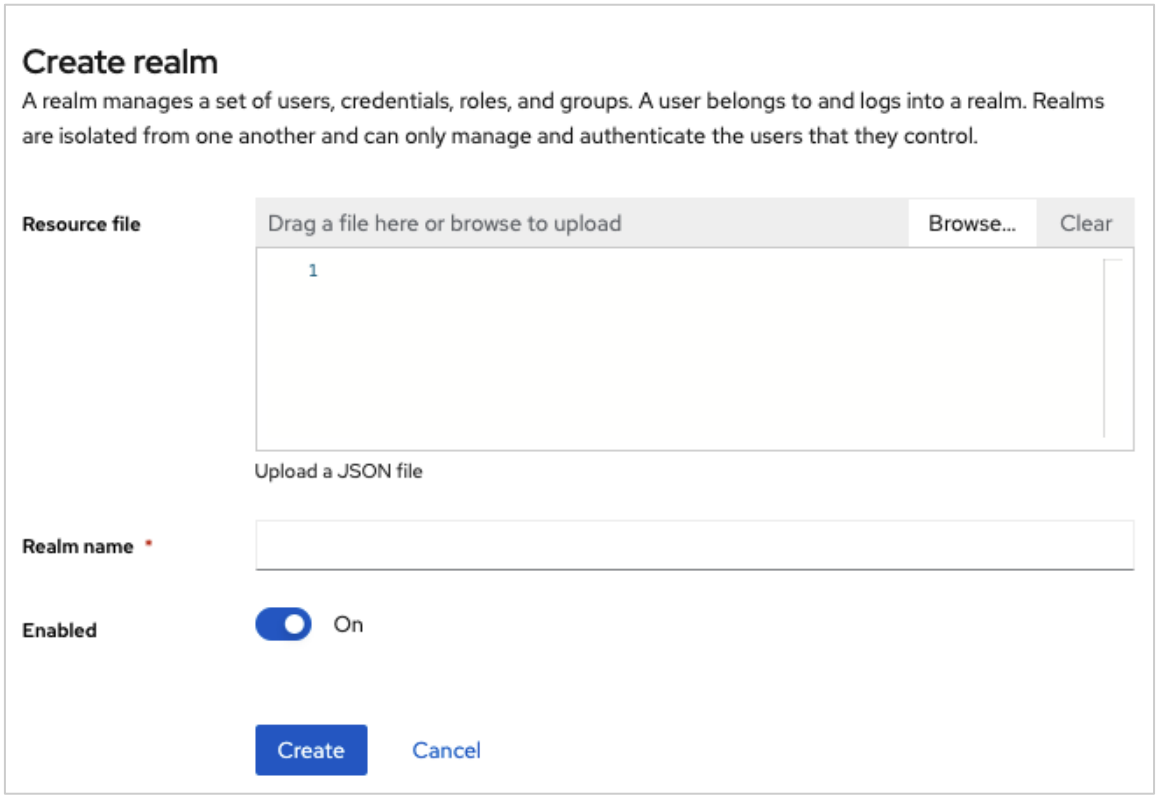
Решая, какие области вам нужны, подумайте о том, какой тип изоляции вы хотите обеспечить для своих пользователей и приложений. Например, вы можете создать область для сотрудников вашей компании и отдельную область для ваших клиентов. Ваши сотрудники войдут в личный кабинет и смогут посещать только внутренние приложения компании. Клиенты будут входить в клиентскую среду и иметь возможность взаимондействовать только с клиентскими приложениями.

Процедура:

- наведите указатель мыши на верхнюю часть левой панели;
- нажмите «Создать область» > Добавить меню областей;



- введите имя области;
- нажмите Создать > Создать область;



Текущая область теперь установлена на область, которую вы только что создали. Вы можете переключаться между мирами, щелкнув имя региона в меню.

12.2.3 Создание пользователей

Необходимо создавать пользователей в той области, которую вы выделили для приложения, необходимые этим пользователям. Избегайте создания пользователей в главной области, которая предназначена только для создания других областей.

Предварительное условие:

- Вы находитесь в области, отличной от области мастера.

Процедура:

- Нажмите «Пользователи» в меню.
- Нажмите «Добавить пользователя».
- Введите данные нового пользователя
  - Имя пользователя — единственное обязательное поле.
- Нажмите «Сохранить». После сохранения сведений откроется страница управления новым пользователем.

Более подробная документация по свойствам пользователей можно найти на официальном сайте по ссылке:

https://www.keycloak.org/docs/latest/server_admin

12.3 Мониторинг работы компонентов Системы

Самое быстрое это проверяем доступность web ui. Если она доступна - все ок. Иначе заходим в консоль по ssh доступу на сервер и проверяем работу службы:

```
# systemctl status keycloak.service
```

Для остановки сервиса необходимо выполнить команду:

```
# systemctl stop keycloak.service
```

Для запуска сервиса необходимо выполнить команду:

```
# systemctl start keycloak.service
```

12.4 Логирование компонентов Системы

Для данной компоненты включение логирования происходит путем передачи параметров в команду запуска.

Параметры логирования, их типы и настройки ротирования описаны в официальной документации по ссылке https://www.keycloak.org/server/logging.

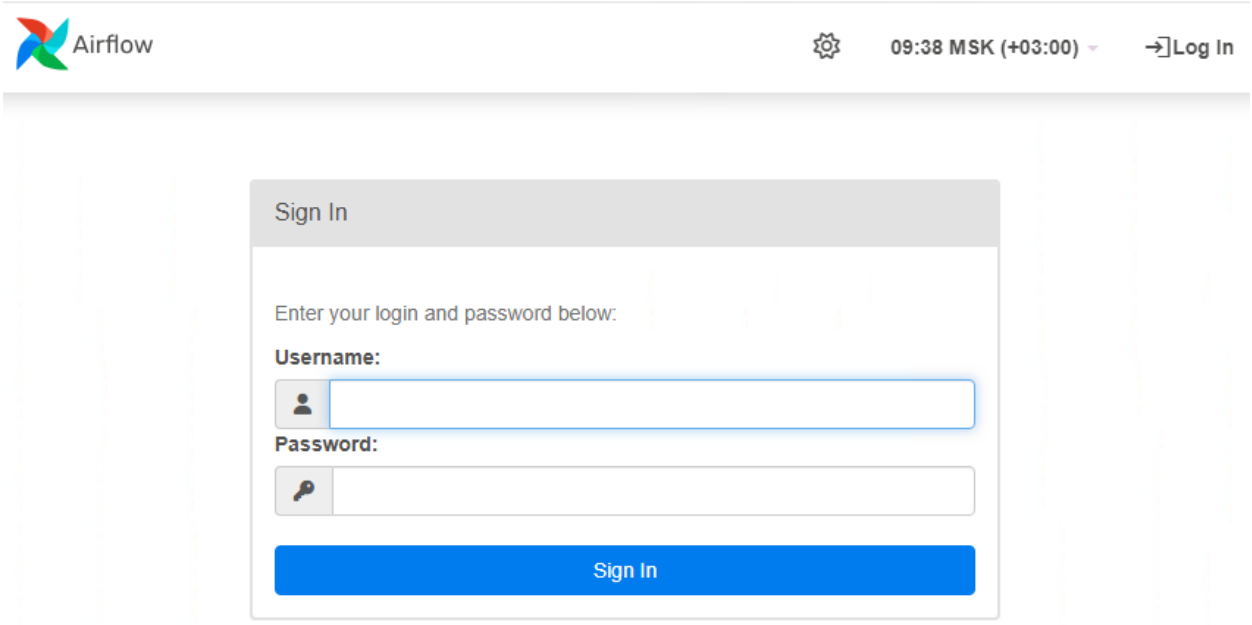
Выбранный метод необходимо будет вписать в systemd unit описанный в руководстве по развертыванию данной компоненты.

13 Администрирование Apache Airflow

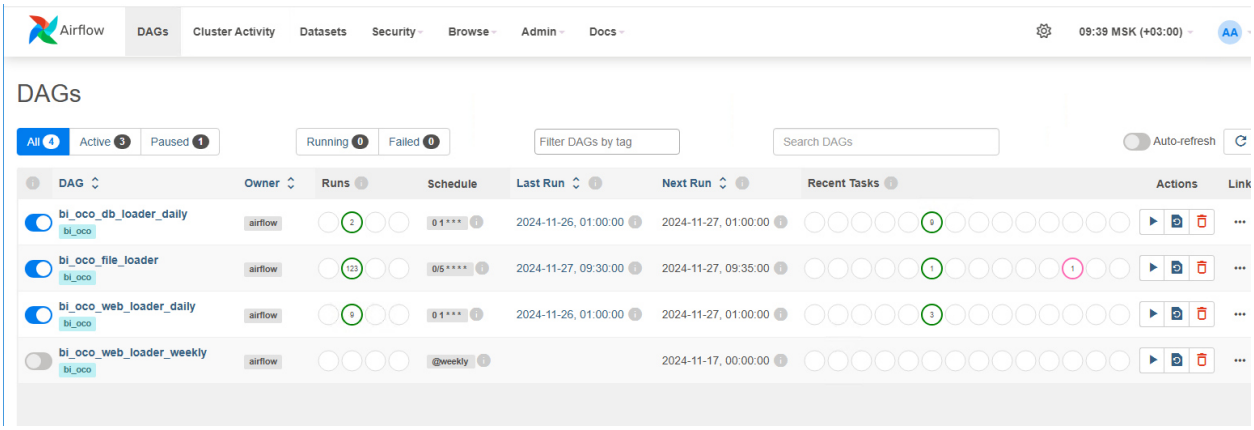
13.1 Доступ к компонентам Системы

Компоненты Apache Airflow можно посмотреть по Web-UI. Для того чтобы открыть компоненты через Web-UI необходимо перейти по ссылке https://sr-dc-0226.corp.gidroogk.com.

После открытия данного адреса необходимо будет пройти авторизацию:

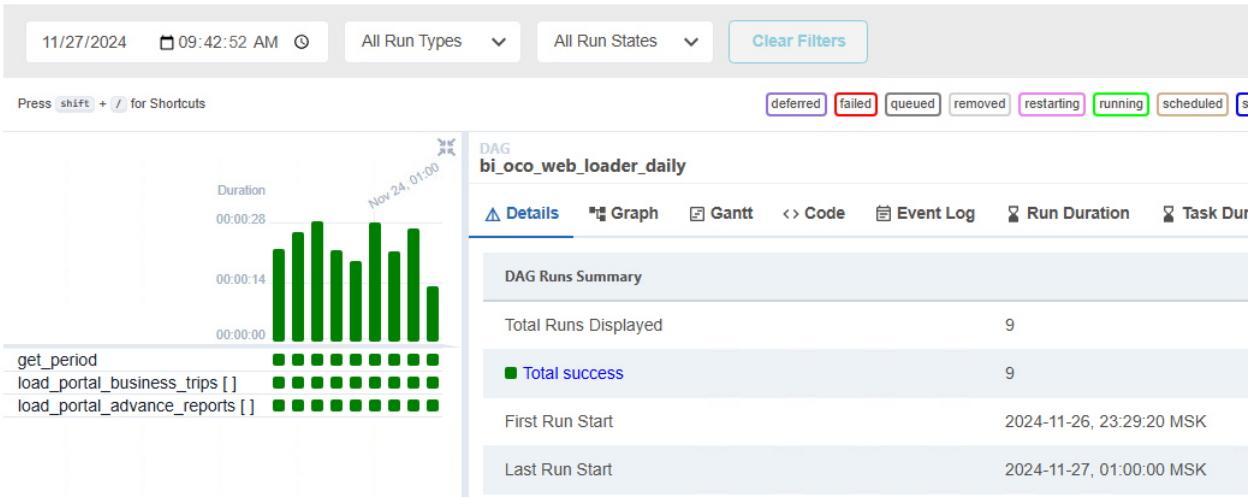


После успешной авторизации вы увидите в браузере окно со всей информацией о всех процессах, запущенных или остановленных в системе, например:



13.2 Мониторинг работы компонентов Системы

Мониторинг работы основных систем компоненты можно наблюдать через основное меню. Есть возможность смотреть работу отдельно каждого процесса:



Для проверки работоспособности сервисов необходимо перейти на сервер по ssh и вызвать команду для проверки состояния работы сервиса. Т.к. сервис работает в докере, то мы будем проверять статус командами докера:

```
sudo docker compose -f /opt/airflow/compose.yaml ps
```

Остановка сервиса производится командой:

```
sudo docker compose -f /opt/airflow/compose.yaml down
```

Запуск сервиса производиться командой:

```
sudo docker compose -f /opt/airflow/compose.yaml up -d
```

13.3 Логирование компонентов Системы

Основные логи работы компоненты можно просмотреть через команды докера:

```
sudo docker compose -f /opt/airflow/compose.yaml --ansi=always logs airflow-scheduler | less -R

sudo docker compose -f /opt/airflow/compose.yaml --ansi=always logs airflow-triggerer | less -R

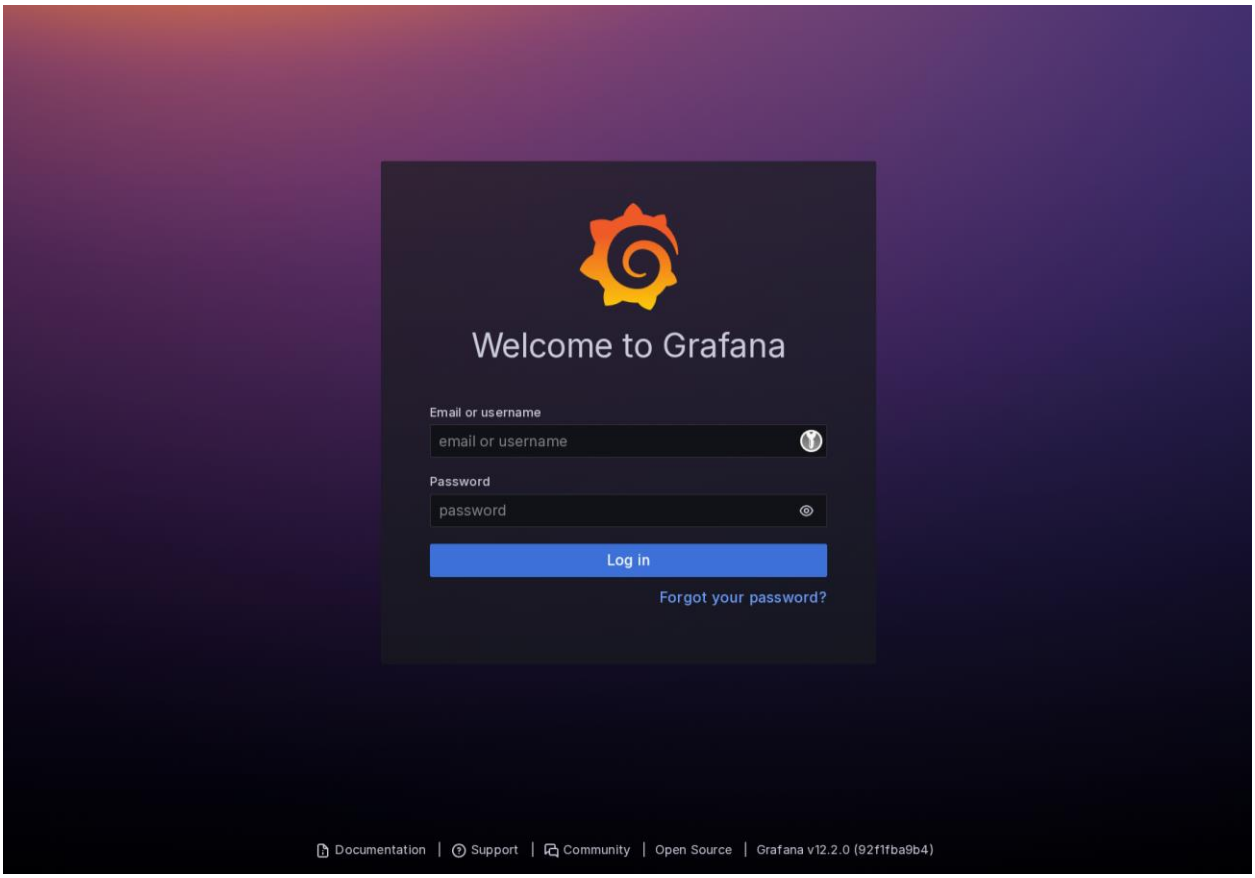
sudo docker compose -f /opt/airflow/compose.yaml --ansi=always logs airflow-webserver | less -R

sudo docker compose -f /opt/airflow/compose.yaml --ansi=always logs airflow-worker | less -R

sudo docker compose -f /opt/airflow/compose.yaml --ansi=always logs redis | less -R
```

14 Администрирование Grafana Loki

14.1 Доступ к компонентам подсистемы



Компоненты Grafana Loki доступны через Web-интерфейс Grafana. Для входа в систему необходимо открыть в браузере ссылку: http://<host>:11080/login:

После открытия данного адреса потребуется пройти авторизацию. Используйте учетные данные администратора, выданные при установке или передаче системы в эксплуатацию.

После успешной авторизации в браузере откроется главная панель Grafana, на которой отображается сводная статистическая информация по логам, поступающим от различных систем.

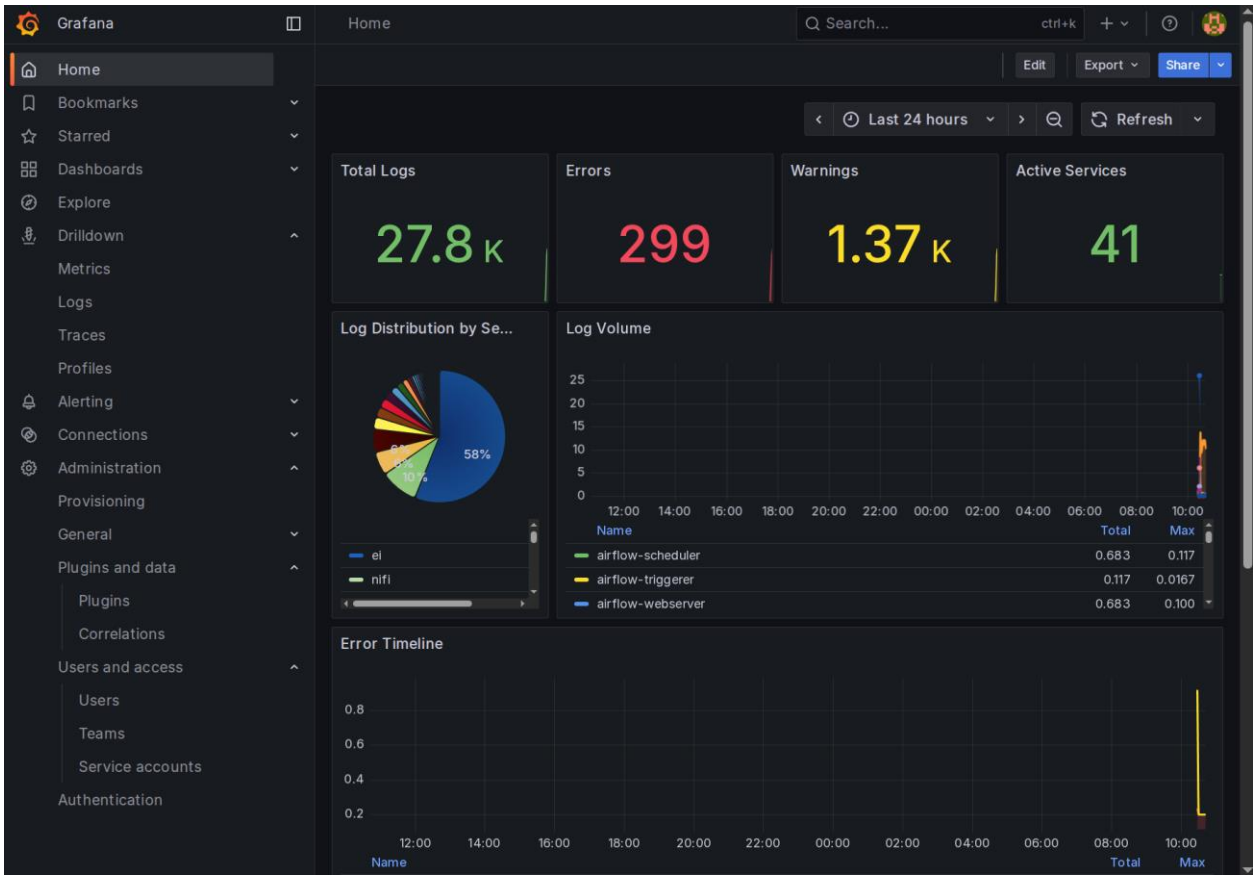
На стартовом дашборде обычно отображаются следующие элементы:

- количество полученных логов за текущий и предыдущий периоды;
- распределение логов по источникам (сервисам);
- доля сообщений разных уровней (INFO, WARN, ERROR);
- график активности логирования во времени.

Эта информация позволяет быстро оценить общую нагрузку и состояние логирующей подсистемы без необходимости переходить в ручной просмотр логов.

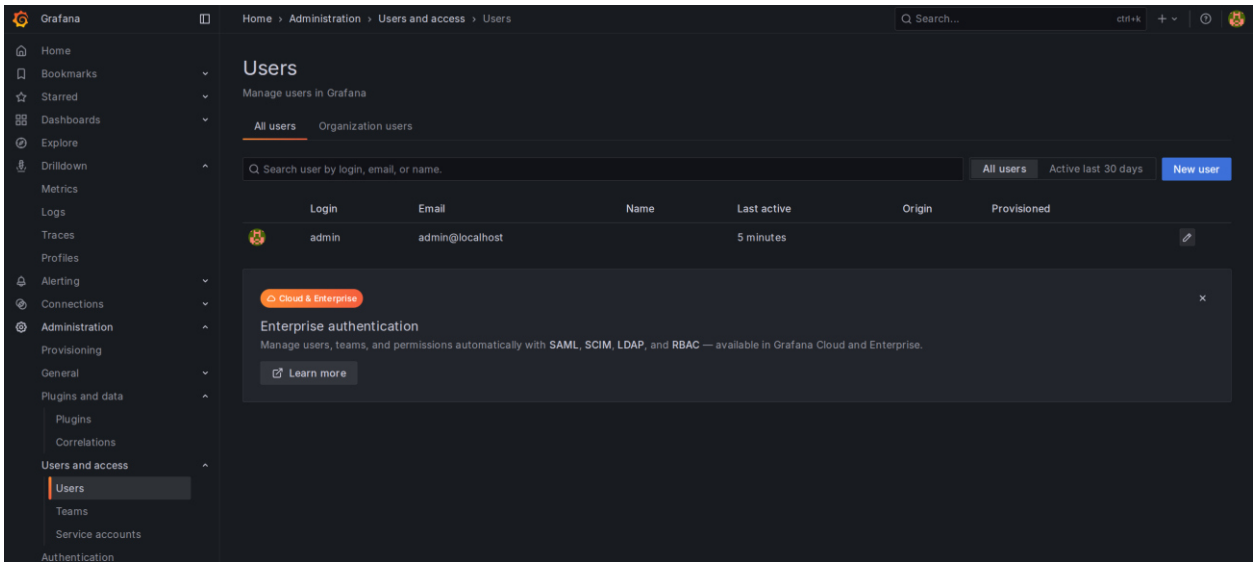
При необходимости дашборд можно отфильтровать по конкретному сервису или диапазону времени.

Пример стартового дашборда отображен на рисунке:



14.2 Управление пользователями Grafana

Управление доступом пользователей выполняется в разделе Administration → Users and access → Users:



В этом разделе меню можно добавлять, изменять и удалять пользователей, а также устанавливать права доступа с помощью указания роли (Viwer / Editor / Admin)

14.3 Мониторинг и управление компонентами подсистемы

Для проверки текущего состояния и управления состоянием docker-контейнеров необходимо подключиться по SSH и выполнить необходимые команды Docker Compose.

Проверка состояния сервисов:

```
sudo docker compose -f /opt/loki/compose.yaml ps
```

Остановка сервисов:

```
sudo docker compose -f /opt/loki/compose.yaml down
```

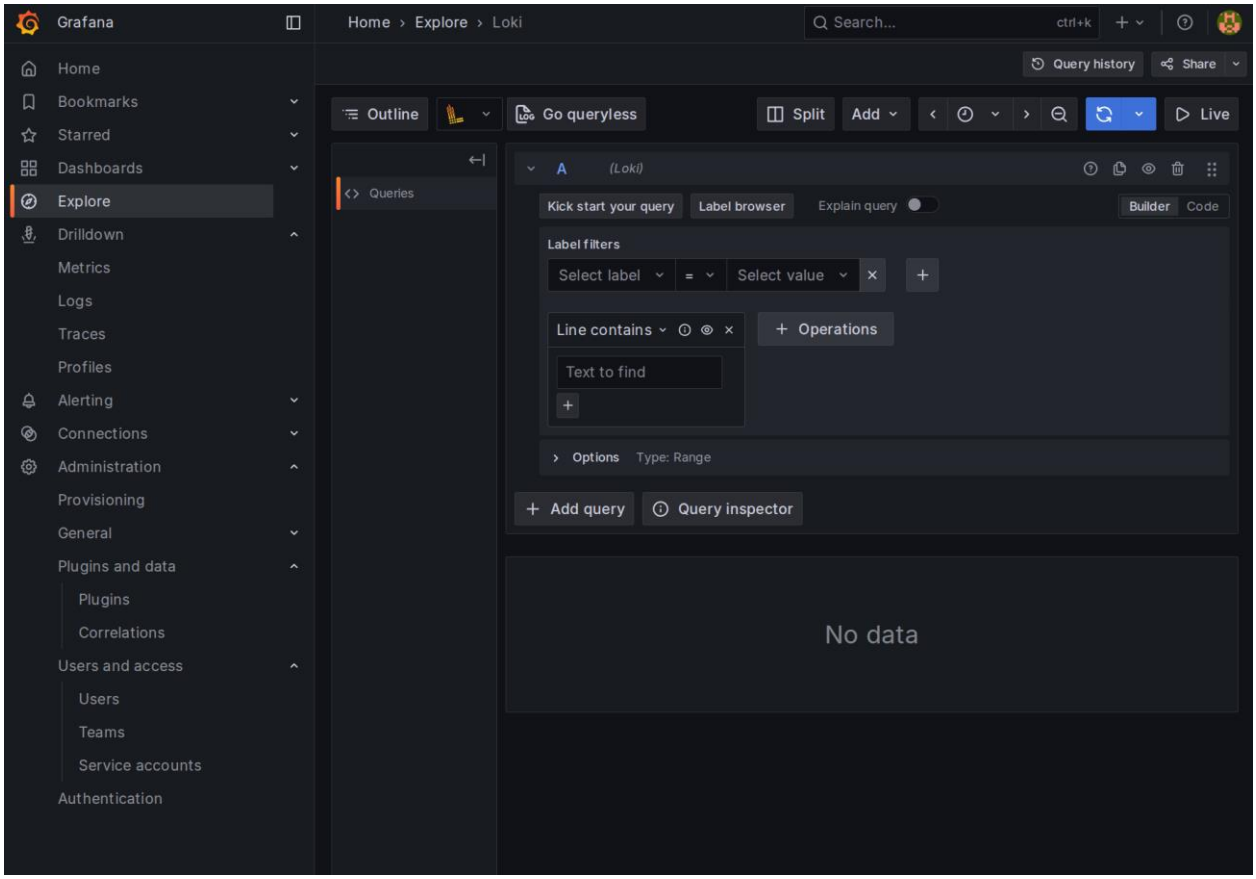
Запуск сервисов:

```
sudo docker compose -f /opt/loki/compose.yaml up -d
```

После перезапуска компонентов рекомендуется открыть Grafana и убедиться, что источник данных Loki активен (раздел Connections → Data sources → Loki).

14.4 Просмотр собираемых из систем логов через Grafana

Мониторинг состояния компонентов Grafana и Loki можно выполнять через веб-интерфейс Grafana. Для этого в разделе Explore → Loki можно наблюдать поток поступающих логов, проверять корректность работы парсинга и фильтрации данных. При этом видна текущая активность, объем данных и задержка получения сообщений. Раздел Explore позволяет выполнять запросы напрямую к Loki и фильтровать сообщения по нужным меткам и значениям:



Пример просмотра логов по конкретному сервису:

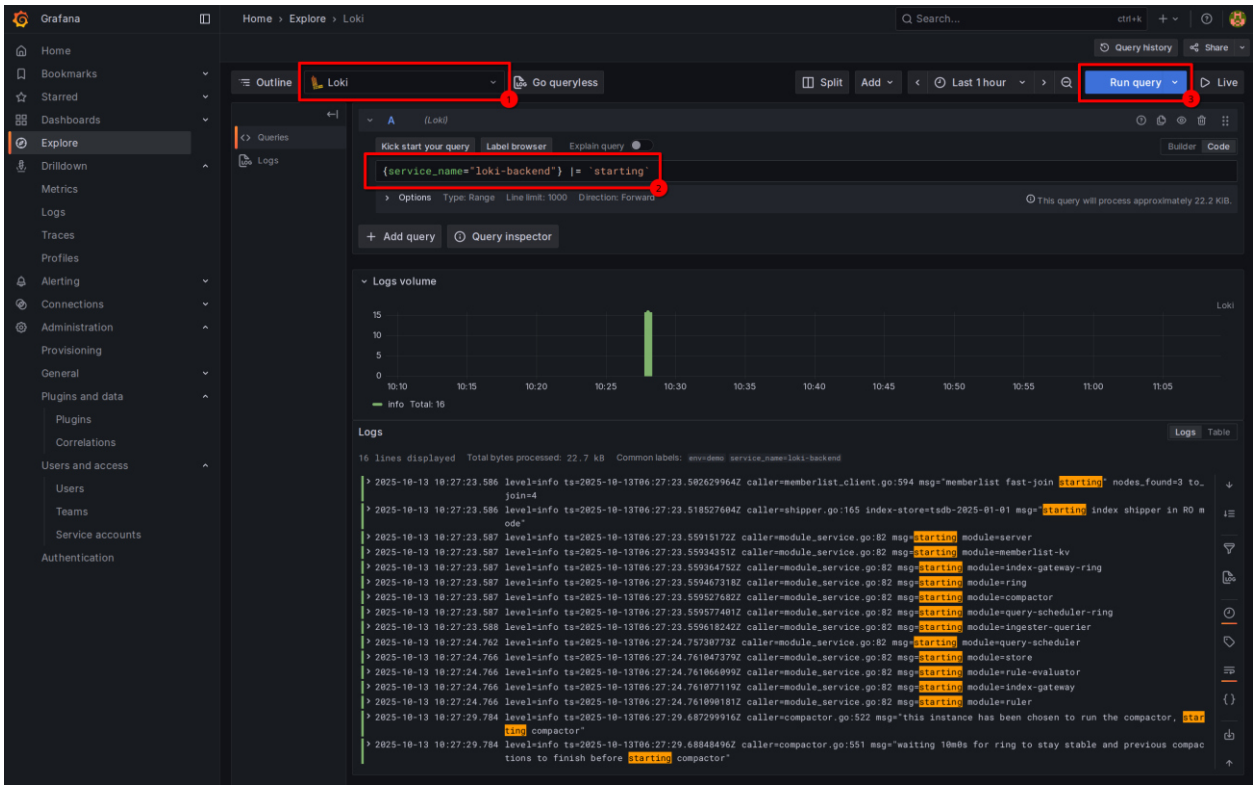
1. Перейдите в основном меню Grafana в пункт Explore.

2. В выпадающем списке Data source выберите источник данных Loki.

3. В поле запроса введите фильтр по метке (будут отбираться сообщения системы loki-backend, в которых встречается текст «starting»), например:

```
{service_name="loki-backend"} |= `starting`
```

4. Нажмите кнопку Run query:



В нижней части окна появится поток логов, соответствующих указанному фильтру. При необходимости можно изменить диапазон времени в правом верхнем углу окна, чтобы просмотреть логи только за интересующий период.

14.5 Логирование компонентов подсистемы через Docker

Кроме просмотра логов самой подсистемы Grafana / Loki через интерфейс Grafana, администратор может напрямую получать логи контейнеров с помощью Docker Compose.

Просмотр логов отдельно по контейнерам:

```
sudo docker compose -f /opt/loki/compose.yaml --ansi=always logs loki | less -R

sudo docker compose -f /opt/loki/compose.yaml --ansi=always logs grafana | less -R
```

Просмотр последних логов всех контейнеров:

```
sudo docker compose -f /opt/loki/compose.yaml logs --tail=100 loki
```

Просмотр состояния всех контейнеров Docker:

```
sudo docker ps -a
```

15 Администрирование TestY

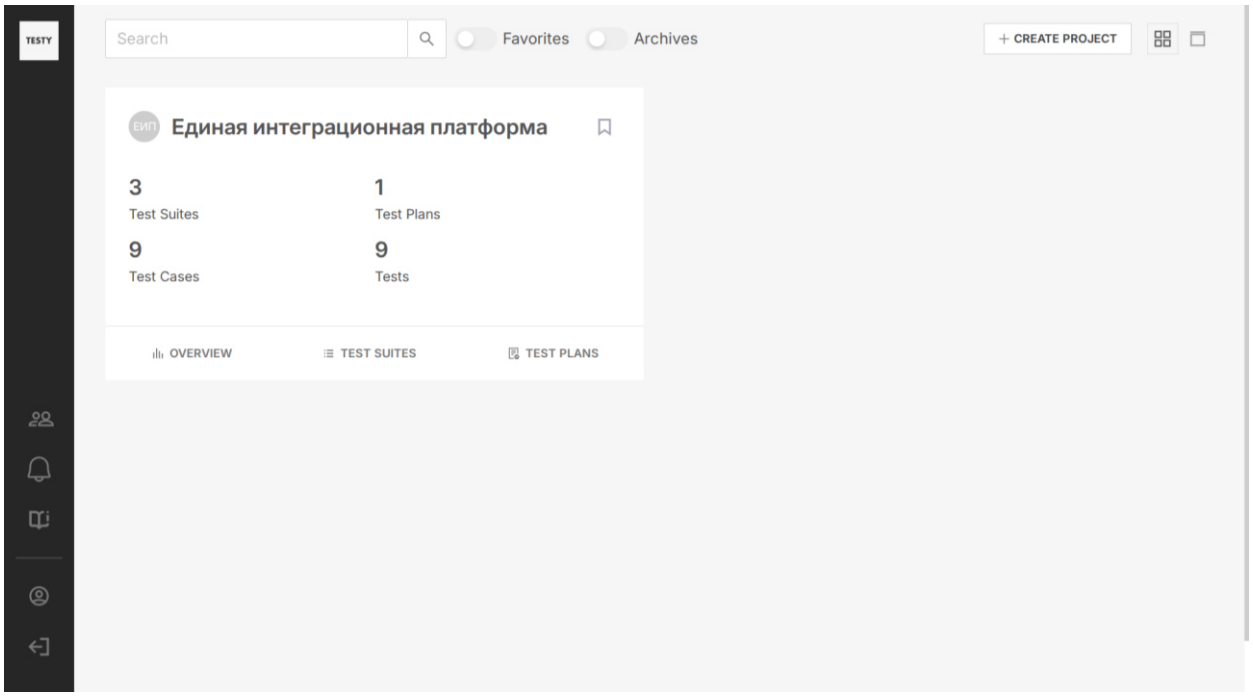
15.1 Доступ к компонентам подсистемы



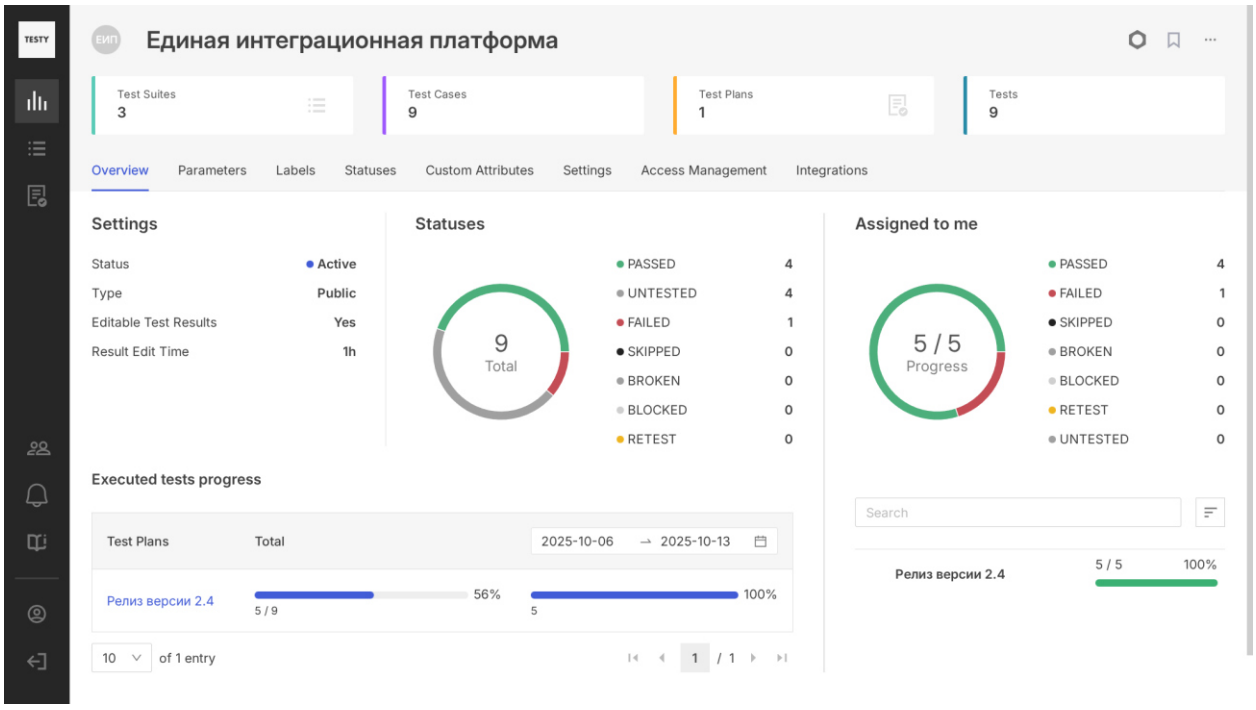
Администрирование TestY выполняется через Web-интерфейс. Для входа в него необходимо открыть в браузере ссылку https://<host>:12443:

После перехода по данному адресу потребуется выполнить авторизацию. Для входа используйте административные учетные данные, указанные при установке системы или её вводе в эксплуатацию.

После успешного входа в систему в браузере откроется главное окно TestY:

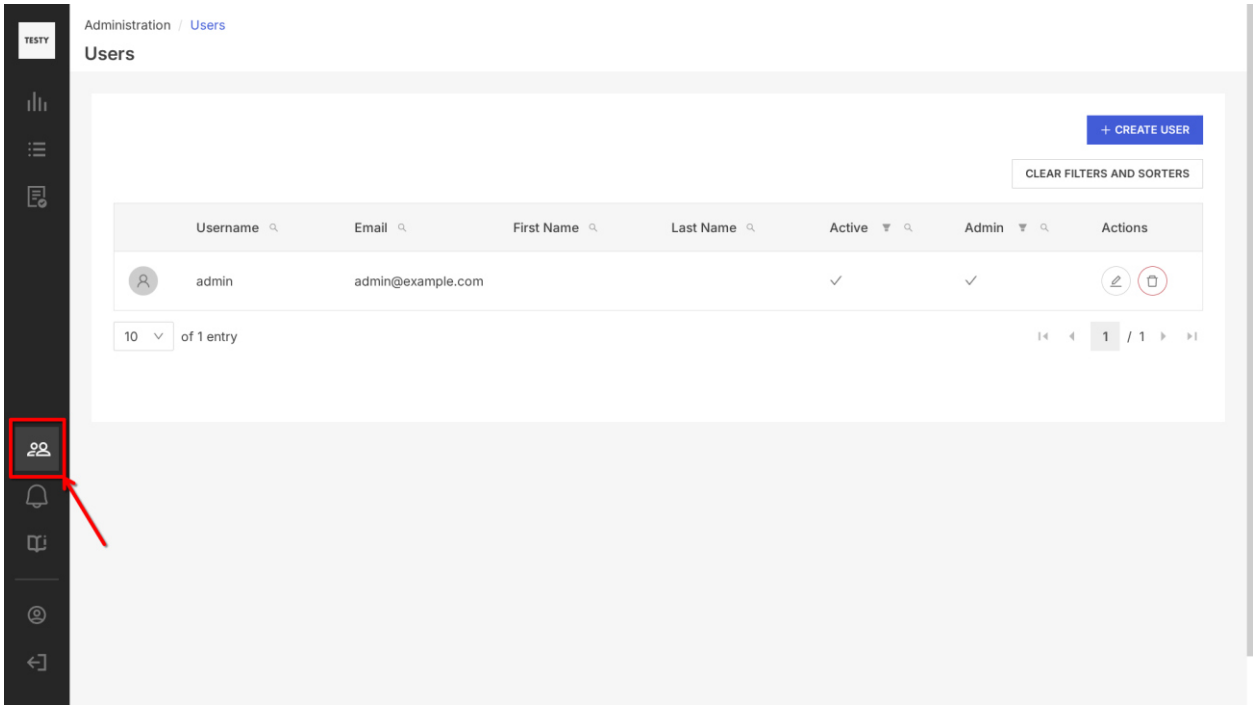


В этом окне отображается список внесенных в систему проектов с общей информацией по ним: количество тестовых наборов, тестов, а также тестовых планов. Детальную информацию о проекте можно увидеть, кликнув по его названию:



15.2 Управление пользователями TestY

Управление доступом пользователей к подсистеме выполняется в разделе Administration → Users:



В этом разделе меню можно добавлять, изменять и удалять пользователей, а также включать / выключать административные права (переключатель Admin).

15.3 Мониторинг и управление компонентами подсистемы

Для проверки текущего состояния и управления состоянием docker-контейнеров необходимо подключиться к соответствующей машине по SSH и в терминале инициировать требуемые действия командами.

Проверка состояния сервисов:

```
sudo docker compose -f /opt/testy/compose.yaml ps
```

Остановка сервисов:

```
sudo docker compose -f /opt/testy/compose.yaml down
```

Запуск сервисов:

```
sudo docker compose -f /opt/testy/compose.yaml up -d
```

15.4 Логирование компонентов подсистемы

Для просмотре логов подсистемы необходимо подключиться к соответствующей машине по SSH и в терминале инициировать требуемые действия командами.

Просмотр логов отдельно по контейнерам:

```
sudo docker compose -f /opt/testy/compose.yaml --ansi=always logs redis |
less -R

sudo docker compose -f /opt/testy/compose.yaml --ansi=always logs pgbouncer |
less -R

sudo docker compose -f /opt/testy/compose.yaml --ansi=always logs celery |
less -R

sudo docker compose -f /opt/testy/compose.yaml --ansi=always logs backend |
```

```
less -R

sudo docker compose -f /opt/testy/compose.yaml --ansi=always logs
notification-worker | less -R

sudo docker compose -f /opt/testy/compose.yaml --ansi=always logs nginx |
less -R
```

Просмотр последних логов всех контейнеров:

```
sudo docker compose -f /opt/testy/compose.yaml logs --tail=100
```

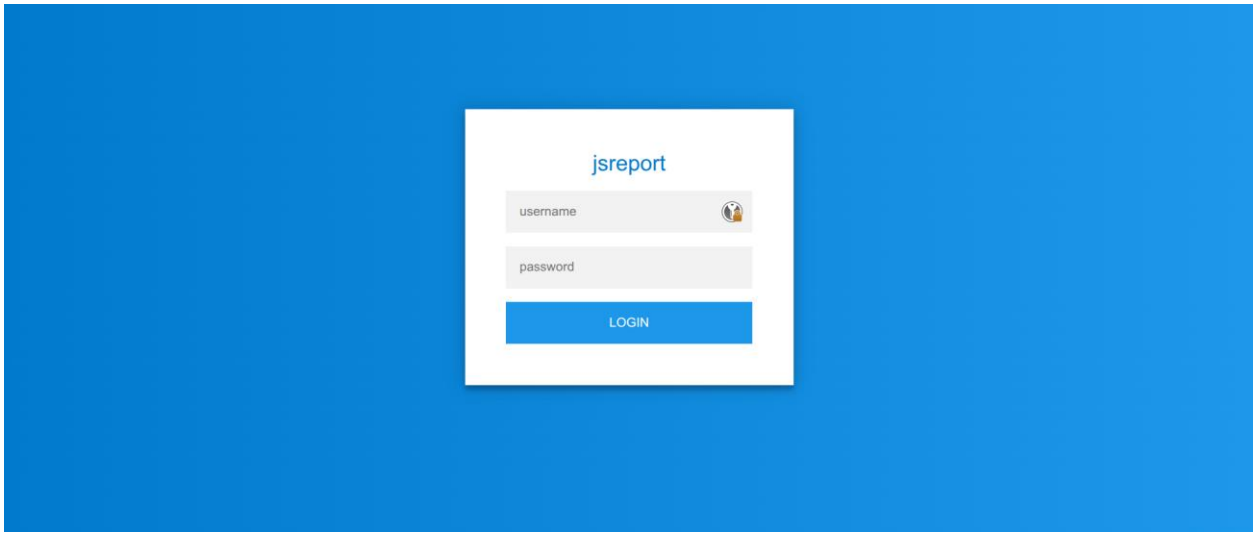
Просмотр состояния всех контейнеров Docker:

```
sudo docker ps -a
```

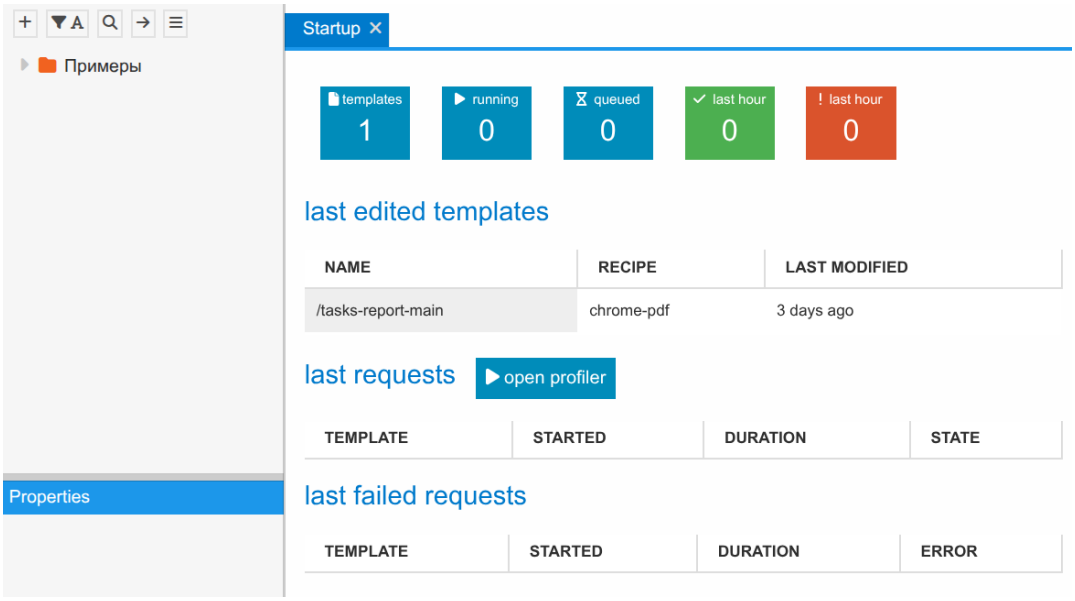
16 Администрирование JSReport

16.1 Доступ к компонентам подсистемы

Администрирование JSReport выполняется через Web-интерфейс. Для входа в него необходимо открыть в браузере ссылку http://<host>:13080:

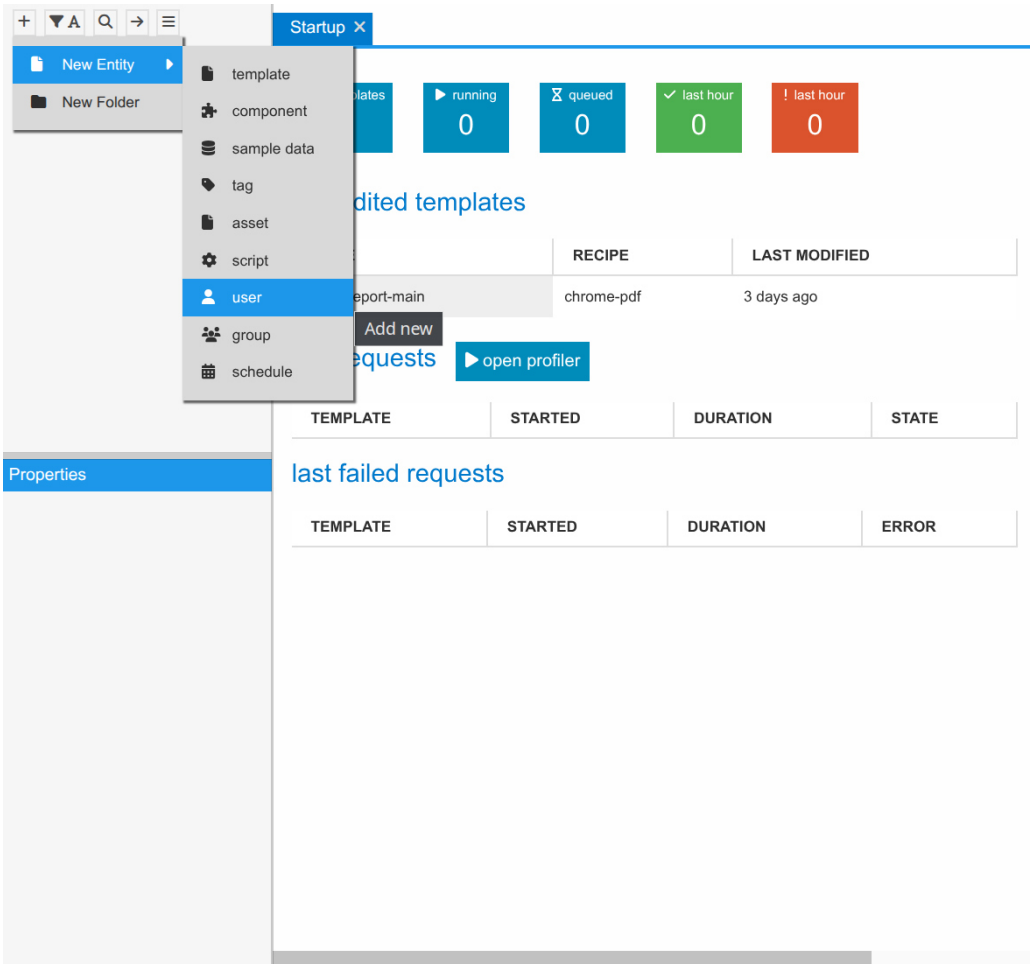


После перехода по этому адресу откроется форма авторизации. Введите учетные данные администратора, установленные при вводе системы в эксплуатацию. После успешной авторизации в браузере появится главное окно JSReport. В этом окне отобразится иерархический список созданных шаблонов и общая информация по шаблонам и отчетам: количество созданных шаблонов, количество сформированных по ним отчетов и количество вызовов API:

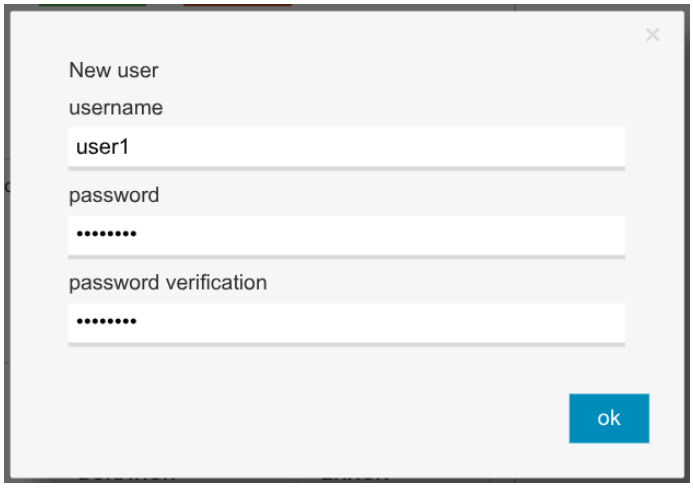


16.2 Управление пользователями JSReport

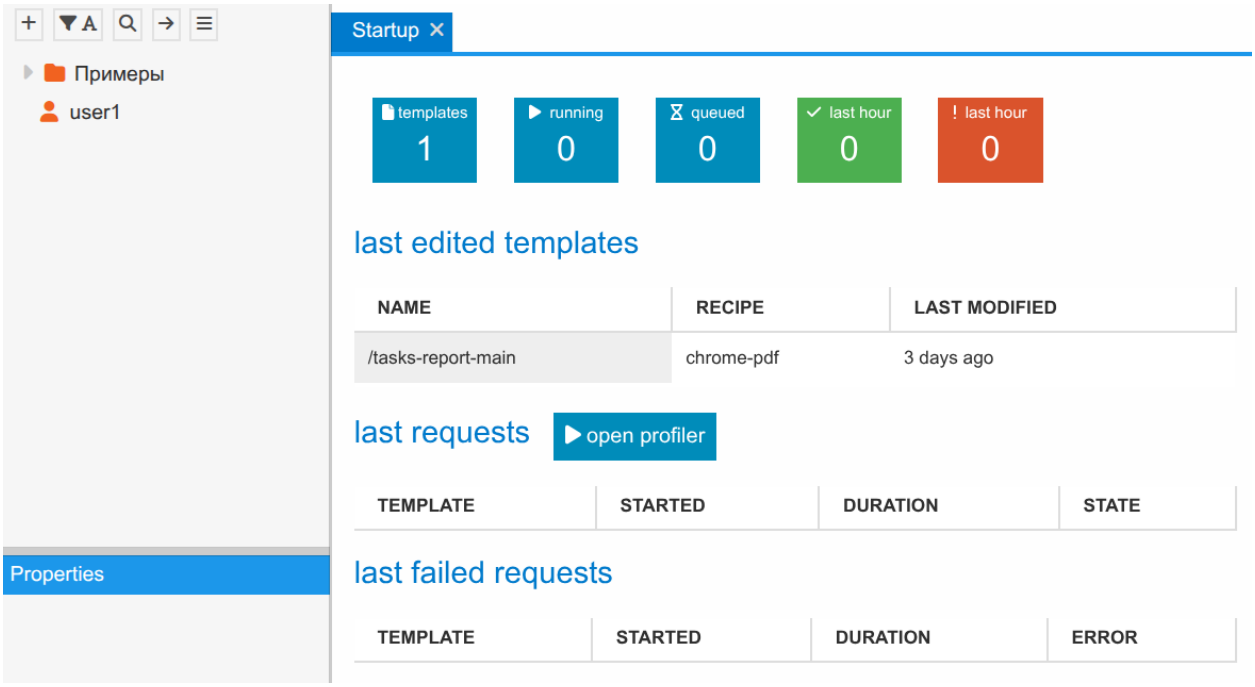
Добавление нового пользователя выполняется выбором пукта меню + → New entity → user:



При выборе этого пункта меню отображается окно, в котором нужно указать имя пользователя и пароль (дважды):



После создания пользователя, он появится в иерархическом списке слева:



Действия с существующими пользователями выполняются через меню по правой кнопке мыши на конкретном пользователе. Доступны действия Переименование, Удаление, Копирование и Просмотр свойств.

Пользователей можно объединять в группы.

Настройка доступа пользователям к конкретным отчетам выполняется указанием разрешенных пользователей в свойствах самих отчетов.

16.3 Мониторинг и управление компонентами подсистемы

Для проверки текущего состояния и управления состоянием docker-контейнеров необходимо подключиться к соответствующей машине по SSH и в терминале инициировать требуемые действия командами.

Проверка состояния сервисов:

```
sudo docker compose -f /opt/jsreport/compose.yaml ps
```

Остановка сервисов:

```
sudo docker compose -f /opt/jsreport/compose.yaml down
```

Запуск сервисов:

```
sudo docker compose -f /opt/jsreport/compose.yaml up -d
```

16.4 Логирование компонентов подсистемы

Для просмотре логов подсистемы необходимо подключиться к соответствующей машине по SSH и в терминале инициировать требуемые действия командами.

Просмотр логов отдельно по контейнерам:

```
sudo docker compose -f /opt/jsreport/compose.yaml --ansi=always logs redis |
less -R
```

Просмотр последних логов всех контейнеров:

```
sudo docker compose -f /opt/jsreport/compose.yaml logs --tail=100
```

Просмотр состояния всех контейнеров Docker:

```
sudo docker ps -a
```

17 Резервное копирование баз данных и компонентов Системы

17.1 Общие рекомендации к резервированию

Таблица 2. Резервное копирование серверов

Сервер	Ресурс	Расписание	Тип резервного копирования	Количество точек восстановления для ВМ
sr-dc-0052 sr-dc-0054 sr-dc-0055 sr-dc-0056 sr-dc-0057 sr-dc-0058 sr-dc-0043 sr-dc-0045 sr-dc-0046 sr-dc-0047 sr-dc-0048 sr-dc-0049 sr-dc-0142 sr-dc-00143 sr-dc-0222 sr-dc-0226	Виртуальная машина	Ежедневно	Копия ВМ	28
sr-dc-0043 sr-dc-0052 sr-dc-0142 sr-dc-0143 (БД rh_db)	Бэкап БД rh_db согласно разделу 4.1.	Совместно с копией ВМ	Полный	28
sr-dc-0049 sr-dc-0058 (БД openmetadatadb)	Бэкап БД Openmetadata согласно разделу 4.2.	Совместно с копией ВМ	Полный	28

Таблица 3 - Резервное копирование БД

Имя сервера	Тип резервного копирования	Расписание	Глубина хранения (количество копий)
sr-dc-0053 sr-dc-0044 БД: supersetdb keycloakdb airflowdb	Полное (FULL)	Каждое воскресенье в 21.00 МСК	5 полных копий, либо не менее 30 дней
	Страничное (PAGE)	С понедельника по субботу в 21:00 МСК	Все копии, начиная с самой ранней полного бэкапа
	Журнал предзаписи (WAL)	Постоянно	Все сегменты, начиная с самого раннего полного бэкапа в цепочке РК.

17.2 Резервное копирование и восстановление данных ClickHouse

Ниже описан автоматизированный способ копирования ClickHouse с использованием утилиты clickhouse-backup.

17.2.1 Установка утилиты clickhouse-backup

```
wget https://github.com/Altinity/clickhouse-backup/releases/download/v2.4.8/clickhouse-backup-2.4.8-1.x86_64.rpm
sudo rpm -ivh clickhouse-backup-2.4.8-1.x86_64.rpm
```

17.2.2 Настройка clickhouse-backup через config файл

По умолчанию файл конфигурации находится по адресу /etc/clickhouse-backup/config.yml, но его можно переопределить с помощью CLICKHOUSE_BACKUP_CONFIG.

Все параметры могут быть перезаписаны через переменные среды. Используйте clickhouse-backup print-config для печати текущей конфигурации.

Просмотрите все элементы конфигурации по умолчанию:

```
clickhouse-backup default-config
```

Создайте директорию хранения конфигов:

```
mkdir -p /etc/clickhouse-backup/
```

Укажите параметры для подключения к Clickhouse:

```
cat > /etc/clickhouse-backup/config.yml << EOF
clickhouse:
username: default
password: "you_password"
host: localhost
port: 9000
data_path: "/var/lib/clickhouse"
general:
remote_storage: none
backups_to_keep_local: 7
backups_to_keep_remote: 31
EOF
```

После указания значения сервера проверьте подключение к Clickhouse через запрос:

```
clickhouse-backup tables
```

Команда выведет список таблиц из Clickhouse. Если список есть, значит конфиг корректный.

Если вы настраиваете локальное резервное копирование, добавьте в конфиг etc/clickhouse-backup/config.yml дополнительную строку: remote_storage: none. Она говорит о том, что облачное хранилище отсутствует.

В основном ошибка о недоступности облачного хранилища состоит в том, что локально backup собран, но не загружен на в облачное хранилище.

```
Local backups:
'backup' (created at 16-07-2021 09:21:38) Remote backups: 2021/07/16 09:24:59
InvalidParameter: 1 validation error(s) found.
minimum field size of 1, ListObjectsV2Input.Bucket.
```

Clickhouse-backup поддерживает загрузку и загрузку резервных копий из хранилища удаленных объектов (например, S3, GCS или IBM COS).

Например, для AWS S3 измените файл конфигурации /etc/clickhouse-backup/config.yml:

```
s3:
Access_key:
secret_key: Bucket:
region: us-east-1
Path: "/some/path/in/bucket"
```

После указания параметров хранилища и выполнения резервного копирования данных можно загрузить backup командой:

```
clickhouse-backup upload "backup-name"
```

17.2.3 Создание резервной копии

Для выполнения резервного копирование остановка СУБД не требуется.

Создайте полную резервную копию БД:

```
clickhouse-backup create
```

Резервная копия хранится в каталоге /var/lib/clickhouse/backup. По умолчанию имя резервной копии — это временная метка. Можно вручную указать имя резервной копии, например,

```
clickhouse-backup create test.backup-19072021
```

Резервная копия содержит два каталога:

- каталог 'metadata': содержит DDL SQL, необходимый для воссоздания;
- 'shadow' каталог: содержит данные, полученные в результате операций ALTER TABLE ... FREEZE.

Резервное копирование одной таблицы выполняется по команде:

```
clickhouse-backup create [-t, --tables=<db>.<table>] <backup_name>
```

Резервное копирование нескольких таблиц выполняется по команде:

```
clickhouse-backup create  -t caihao.test_restore_tab,caihao.ch1 ch_bak_2tab
```

Просмотрите файлы резервных копий по команде:

```
clickhouse-backup list
```

17.2.4 Восстановление данных из резервной копии

Восстановление из резервной копии происходит с помощью команды restore. Некоторые параметры:

- --table - восстанавливает только определенные таблицы. Например, для конкретной базы данных: --table = dbname.
- --schema -восстанавливает только структуру таблицы;
- --data – восстанавливает только данные.

Пример:

```
NAME:
   clickhouse-backup restore - Create schema and restore data from backup


USAGE:
   clickhouse-backup restore  [-t, --tables=<db>.<table>] [-m, --restore-
database-mapping=<originDB>:<targetDB>[,<...>]] [--
partitions=<partitions_names>] [-s, --schema] [-d, --data] [--rm, --drop] [-
i, --ignore-dependencies] [--rbac] [--configs] <backup_name>


OPTIONS:
   --config value, -c value                    Config 'FILE' name. (default:
"/etc/clickhouse-backup/config.yml") [$CLICKHOUSE_BACKUP_CONFIG]
   --table value, --tables value, -t value     Restore only database and
objects which matched with table name patterns, separated by comma, allow ?
and * as wildcard
   --restore-database-mapping value, -m value  Define the rule to restore
data. For the database not defined in this struct, the program will not deal
with it.
   --partitions partition_id                   Restore backup only for
selected partition names, separated by comma
if PARTITION BY clause returns numeric not hashed values for partition_id
field in system.parts table, then use --
partitions=partition_id1,partition_id2 format
if PARTITION BY clause returns hashed string values, then use --
partitions=('non_numeric_field_value_for_part1'),('non_numeric_field_value_fo
r_part2') format
if PARTITION BY clause returns tuple with multiple fields, then use --
partitions=(numeric_value1,'string_value1','date_or_datetime_value'),(...)
format
values depends on field types in your table, use single quote for String and
Date/DateTime related types
look to system.parts partition and partition_id fields for details
https://clickhouse.com/docs/en/operations/system-tables/parts/
   --schema, -s                                Restore schema only
   --data, -d                                  Restore data only
   --rm, --drop                                Drop exists schema
objects before restore
```

```
   -i, --ignore-dependencies                              Ignore dependencies
when drop exists schema objects
   --rbac, --restore-rbac, --do-restore-rbac              Restore RBAC related
objects
   --configs, --restore-configs, --do-restore-configs  Restore 'clickhouse-
server' CONFIG related files
```

17.2.5 Создание резервной копии по расписанию через crontab

Скрипт cron для ежедневного резервного копирования и удаленной загрузки:

```
#!/bin/bash
BACKUP_NAME=my_backup_$(date -u +%Y-%m-%dT%H-%M-%S)
clickhouse-backup create $BACKUP_NAME >> /var/log/clickhouse-backup.log 2>&1
exit_code=$?
if [[ $exit_code != 0 ]]; then
  echo "clickhouse-backup create $BACKUP_NAME FAILED and return $exit_code
exit code"
  exit $exit_code
fi

clickhouse-backup upload $BACKUP_NAME >> /var/log/clickhouse-backup.log 2>&1
exit_code=$?
if [[ $exit_code != 0 ]]; then
  echo "clickhouse-backup upload $BACKUP_NAME FAILED and return $exit_code
exit code"
  exit $exit_code
fi
```

При добавлении скрипта в /etc/crontab, проверьте, что в переменной PATH указан путь, где лежит бинарный файл clickhouse-backup, в противном случае скрипт выполнен не будет.

```
PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/bin
```

17.3 Резервное копирование и восстановление данных OpenMetadata

17.3.1 Требования

Backup CLI необходимо использовать с openmetadata-ingestion версией 0.13.1 или выше. Однако мы рекомендуем убедиться, что вы всегда используете последнюю openmetadata-ingestion версию, чтобы все улучшения были реализованы в CLI.

17.3.2 Установка

Backup CLI входит в базовый openmetadata-ingestion пакет Python. Вы можете установить его с помощью:

```
pip install openmetadata-ingestion
```

17.3.3 Предварительные работы

Это пользовательская утилита. Поскольку почти все таблицы содержат GENERATED столбцы, прямое использование mysqldump или pg_dump не является надежным, поскольку для получения правильных данных потребуются некоторые дополнительные шаги по очистке.

Вместо этого необходимо использовать утилиту, которая просто выгружает необходимые данные.

Требованием для запуска процесса является то, что в целевой базе данных должна быть выполнена миграция Flyway.

Утилита резервного копирования сгенерирует файл SQL, который будет выполнять две функции:

1) TRUNCATE таблиц OpenMetadata
2) INSERT данных, которые были сохранены

Затем вы можете запустить данный скрипт для восстановления данных.

Кроме того, убедитесь, что в базе данных назначения восстановления еще нет данных OpenMetadata, а если они есть, то они будут заменены скриптом SQL.

17.3.4 Создание резервной копии через backup CLI

После успешной установки можно рассмотреть различные варианты запуска Backup CLI:

```
> metadata backup -h

usage: metadata backup [-h] -H HOST -u USER -p PASSWORD -d DATABASE [--port
PORT] [--output OUTPUT]
                       [--upload-destination-type {AWS,AZURE}] [--upload
UPLOAD UPLOAD UPLOAD] [-o OPTIONS] [-a ARGUMENTS]
                       [-s SCHEMA]

optional arguments:
  -h, --help            show this help message and exit
  -H HOST, --host HOST  Host that runs the database
  -u USER, --user USER  User to run the backup
  -p PASSWORD, --password PASSWORD
                        Credentials for the user
  -d DATABASE, --database DATABASE
                        Database to backup
  --port PORT           Database service port
  --output OUTPUT       Local path to store the backup
  --upload-destination-type {AWS,AZURE}
                        AWS or AZURE
  --upload UPLOAD UPLOAD UPLOAD
                        S3 endpoint, bucket & key to upload the backup file
  -o OPTIONS, --options OPTIONS
```

```
  -a ARGUMENTS, --arguments ARGUMENTS
  -s SCHEMA, --schema SCHEMA
```

- Если вы используете MySQL для Metadata то запуск будет выглядеть следующим образом:

```
> python -m metadata backup -u openmetadata_user -p openmetadata_password -H
mysql -d openmetadata_db --port 3306
```

- Если вы используете PostgreSQL, добавьте в команду параметр -s, указывающий schema:

```
> python -m metadata backup -u openmetadata_user -p openmetadata_password -H
postgresql -d openmetadata_db --port 5432 -s public
```

В данных командах есть минимальный набор из четырех обязательных параметров, необходимый для доступа к БД и запуска резервного копирования:

- host;
- user;
- password;
- database

Обратите внимание, что у пользователя должен быть как минимум доступ на чтение к базе данных. По умолчанию команда подключается через порт 3306. Данный порт можно переопределить используется ключ-параметр --port

После успешного выполнения резервного копирования, Backup CLI создаст файл дампа, который выглядит как openmetadata_YYYYmmddHHMM_backup.sql. Такое наименование содержит дату создания каждой резервной копии, что поможет впоследствии в поиске нужной копии. Также в команде резервного копирования можно указать конечный путь, который создастся, если он не существует, через ключ-команду --output

17.3.5 Восстановление Metadata БД

Убедитесь, что при восстановлении метаданных ваш сервер OpenMetadata остановлен. В противном случае могут возникнуть конфликты в кэшированных значениях и конкретных элементах, которые сервер создает во время запуска и которые также будут присутствовать в SQL-файле восстановления.

Дамп файл SQL, созданный с помощью Backup metadata CLI, можно восстановить с помощью Restore metadata CLI.

Существуют различные варианты запуска команды Restore metadata CLI:

```
> metadata restore -h

usage: metadata restore [-h] -H HOST -u USER -p PASSWORD -d DATABASE [--port
PORT] --input INPUT [-o OPTIONS]
                        [-a ARGUMENTS] [-s SCHEMA]


optional arguments:
```

```
  -h, --help            show this help message and exit
  -H HOST, --host HOST  Host that runs the database
  -u USER, --user USER  User to run the restore backup
  -p PASSWORD, --password PASSWORD
                        Credentials for the user
  -d DATABASE, --database DATABASE
                        Database to restore
  --port PORT           Database service port
  --input INPUT         Local backup file path for restore
  -o OPTIONS, --options OPTIONS
  -a ARGUMENTS, --arguments ARGUMENTS
  -s SCHEMA, --schema SCHEMA
```

Пример команды восстановления Postgres БД будет выглядеть так:

```
> python -m metadata restore -u openmetadata_user -p openmetadata_password -H
localhost --port 5432 -d openmetadata_db --input
openmetadata_202309301715_backup.sql
```

В результате успешного выполнения команда выдаст:

```
Restoring OpenMetadata backup for localhost:5432/openmetadata_db...
Backup restored from openmetadata_202309301715_backup.sql
```

18 Настройки производительности системы

18.1 Настройка производительности на уровне ОС

Когда дело доходит до производительности, операционная система на которой работает сервер, играет важную роль. Ниже описаны параметры, которые можно настроить для оптимизации производительности сети и ОС. Необходимо настроить следующие параметры в файле /etc/sysctl.conf ОС Linux для максимальной согласованности системы и организации правил загрузки программ при запуске и работе компьютера. Эти параметры могут быть установлены для указания большего диапазона портов, более эффективного значения времени ожидания соединения TCP и ряда других важных настроек на уровне ОС.

```
net.ipv4.tcp_fin_timeout = 30
fs.file-max = 2097152
net.ipv4.tcp_tw_recycle = 1
net.ipv4.tcp_tw_reuse = 1
net.core.rmem_default = 524288
net.core.wmem_default = 524288
net.core.rmem_max = 67108864
net.core.wmem_max = 67108864
net.ipv4.tcp_rmem = 4096 87380 16777216
net.ipv4.tcp_wmem = 4096 65536 16777216
net.ipv4.ip_local_port_range = 1024 65535
```

Ниже перечислены описание параметров и значения по умолчанию.

Наименование параметра	Описание	Рекомендуемое значение
net.ipv4.tcp_fin_timeout	Этот параметр позволяет настроить промежуток времени (в секундах), который требуется TCP для получения окончательного FIN перед закрытием сокета. Установка этого параметра необходима для предотвращения DoS-атак.	30
net.ipv4.tcp_tw_recycle	Это позволяет быстро утилизировать сокеты TIME_WAIT. Изменяйте этот параметр с осторожностью и только во внутренних сетях, где скорость сетевого подключения выше. Не рекомендуется использовать net.ipv4.tcp_tw_recycle = 1 при работе с трансляцией сетевых адресов (NAT), например, если вы развертываете продукты в EC2 или любой другой среде, настроенной с помощью NAT.	1
net.ipv4.tcp_tw_reuse	Этот параметр позволяет повторно использовать сокеты в состоянии TIME_WAIT для новых соединений, когда это безопасно с точки зрения сетевого стека.	1
net.core.rmem_default	Этот параметр устанавливает размер буфера приема	524288

	ОС по умолчанию для всех типов соединений.	
net.core.wmem_default	Это устанавливает размер буфера отправки ОС по умолчанию для всех типов соединений.	524288
net.core.rmem_max	Этот параметр устанавливает максимальный размер буфера приема ОС для всех типов соединений.	67108864
net.core.wmem_max	Этот параметр устанавливает максимальный размер буфера отправки ОС для всех типов соединений.	67108864
net.ipv4.tcp_rmem	Этот параметр определяет пространство буфера приема для каждого соединения TCP и имеет три значения: Первое значение - это минимальное пространство буфера приема для каждого соединения TCP. Этот буфер всегда выделяется для сокета TCP, даже в высоко нагруженной ситеме. Второе значение - это пространство буфера приема по умолчанию, выделенное для каждого сокета TCP. Это значение переопределяет значение /proc/sys/net/core/rmem_default используемое другими протоколами. Последнее значение - это максимальное пространство буфера приема, выделенное для сокета TCP.	4096 87380 16777216
net.ipv4.tcp_wmem	Этот параметр указывает пространство буфера отправки для каждого TCP-соединения и имеет три значения: Первое значение - это минимальное пространство буфера отправки TCP, доступное для одного сокета TCP. Второе значение - это пространство буфера отправки по умолчанию, разрешенное для использования одним сокетом TCP. Третье значение - это максимальное пространство буфера отправки TCP. Каждый TCP-сокет имеет определенный объем буферного пространства, который необходимо использовать до заполнения буфера, и каждое из трех значений используется в различных условиях.	4096 65536 16777216
net.ipv4.ip_local_port_range	Этот параметр определяет диапазон локальных портов, который используется TCP и UDP для выбора локального порта. Первый номер - это первый локальный порт, разрешенный для трафика TCP и UDP, а второй номер - последний номер порта. В Linux диапазон номеров IP-портов по умолчанию, разрешенный для трафика TCP и UDP, невелик. Вы можете увеличить этот диапазон.	1024 65535
fs.file-max	Этот параметр отражает максимальное количество файловых дескрипторов, которое может выделить ядро. Ядро имеет встроенное ограничение на количество файлов, которые может открыть процесс. Если вам нужно увеличить этот предел, вы можете увеличить значение fs.file-max.	2097152

Для изменения количества разрешенных открытых файлов для пользователей системы, настройки производятся в файле /etc/security/limits.conf. Оптимальное значение этих параметров зависит от окружения.

```
soft nofile 4096
hard nofile 65535
```

Стоит отметить:
Так как эти параметры применяются на уровне ОС, изменение параметров может повлиять на другие программы, работающие на сервере. Указанные значения могут быть не оптимальными для вашей системы. Необходимо применить значения и запустить тест производительности, чтобы найти оптимальные значения производительности системы.

18.2 Настройки JVM

Чтобы повысить производительность системы, можно изменить параметры JVM.

Параметры JVM настраиваются в <EI_HOME>/bin/integrator.sh. Ниже приведены наиболее важные параметры JVM, которые необходимо настроить:

- Максимальное выделение памяти кучи (Xmx) - этот параметр устанавливает максимальную память кучи, выделенную для JVM. Увеличение выделения памяти увеличивает объем памяти, доступной для сервера, что приводит к увеличению максимальной TPS и уменьшению задержки. Рекомендуется выделить не менее 2 ГБ памяти кучи для экземпляров. К примеру, если нужно установить максимальный объем кучи памяти в 2 ГБ, необходимо настроить этот параметр следующим образом:
- -Xms2048m -Xmx2048m
- где Xmx - это максимальный пул выделения памяти для JVM.

Когда элемент XML имеет большое количество подэлементов и система пытается обработать все подэлементы, системе может не хватить памяти. Что приведет к нестабильности системы. Чтобы избежать этой проблемы, можно определить максимальный уровень замен сущностей, который разрешает анализатор XML в системе.

В файле <EI_HOME>/bin/integrator.sh можно задать этот параметр. Предел расширения сущности по умолчанию составляет 64000.

```
-DentityExpansionLimit = 10000
```

18.3 Конфигурация кэширования

Параметр <Cache>, настраиваемый в файле carbon.xml (хранится в каталоге <EI_HOME>/repository/conf), устанавливает глобальное время кэширования артефактов. Параметр указывается в минутах. Это значение указывает период времени, после которого кэш будет обновляться. Если параметр не имеет конкретного значения, то будет применяться значение по умолчанию равное 15 минутам. При изменении данного параметра обязательно требуется перезапуск сервера.

Пример параметра указан ниже:

```
< Cache>
<! - Тайм-аут кэша по умолчанию, в минутах ->
<DefaultCacheTimeout> 15 </ DefaultCacheTimeout>
</ Cache>
```